PMDF System Manager's Guide

Order Number: N-5301-66-NN-N

February 2020

This document describes the configuration and usage of version 6.8 of the PMDF, PMDF-MTA, and PMDF-TLS software.

Revision/Update Information:	This manual supersedes the V6.7 *PMDF System Manager's Guide*
Software Version:	PMDF V6.8
Operating System and Version:	Red Hat Enterprise Linux 7 or later on x86_64; (or other compatible Linux distribution)
	OpenVMS Alpha V7.3-2 or later;
	OpenVMS I64 V8.2 or later;

Process Software, LLC
959 Concord Street
Framingham, MA 01701-4682 USA
Voice: +1 508 879 6994; FAX: +1 508 879 0042
info@process.com

Contents

Volume I

Volume II

Volume III

Volume IV

EXAMPLES

FIGURES

TABLES

Preface

Purpose of This Manual

This manual describes the structure, configuration, and use of PMDF. The intended audience is system managers who want to become familiar with how PMDF operates. In particular, this document describes many customizable parts of PMDF that can be altered to adapt PMDF to a particular environment. The reader is assumed to be quite familiar with networking concepts and the operating system on which PMDF is to be installed.

This manual does not provide a description of PMDF suitable for end users; end users should refer to the appropriate edition of the *PMDF User's Guide*.

Overview of This Manual

This guide is a long and technical document. If you are new to PMDF, you should skim the entire document, skipping the discussion in Part 4 of those PMDF layered products which you will not be installing, as well as the channels described in Chapters 17–26 for which you do not have any use. Once you are somewhat familiar with PMDF you can then perform the installation described in the appropriate edition of the *PMDF Installation Guide*, referring back to this manual as needed.

PMDF is a large and complex package capable of being configured to meet almost any task. If your site is typical of most, the initial configuration generated by the configuration utilities will suffice to get you up and running in a minimal amount of time. After you have an initial configuration, study it and use it as an example as you read the following chapters. As you become comfortable with PMDF, you will find that you want to make changes here and there, modify the behavior of some channels, or even add additional channels to your configuration. Or perhaps you will want to set up some databases, or implement a centralized naming system. By all means do so.[1] PMDF, as you will find, has quite a few knobs and switches which you can manipulate and more often than not you will discover that there are not one but several means of dealing with a given issue. If anything, PMDF is too flexible.

This manual consists of four volumes, together comprising forty-two chapters:

Chapter 1, Structure and Overview, describes the overall structure of PMDF and the components which together form PMDF.

Chapter 2, The Configuration File: Domain Rewrite Rules & the Channel/Host Table, describes the PMDF configuration file, including domain rewriting rules and the channel/host table, channel definitions, and the available channel keywords.

[1] There are, of course, some problems best left alone or for which what seems the obvious solution is not really a good one. When in doubt just ask.

Chapter 3, Aliases, Forwarding, and Centralized Naming, describes the alias file and database, and other means of changing addresses, forwarding mail, and establishing centralized naming schemes, including the directory channel.

Chapter 4, Mailing Lists and MAILSERV, describes mailing lists. It also describes automated message processing via the MAILSERV channel.

Chapter 5, The Mapping File, describes the mapping file.

Chapter 6, Character Set Conversions and Message Reformatting , describes internal message reformatting conversions.

Chapter 7, The PMDF Option File, describes the PMDF option file.

Chapter 8, Maintaining the Configuration, describes how to compile your PMDF configuration information so as to decrease the time required for processing jobs to load configuration information. It also discusses restarting PMDF components after PMDF configuration changes.

Chapter 9, The PMDF Process Symbiont (OpenVMS), describes the PMDF Process Symbiont which is used to schedule and execute PMDF processing jobs through the OpenVMS queuing system.

Chapter 10, The PMDF Job Controller (UNIX and Windows), describes the PMDF Job Controller which is used to control PMDF processing jobs on UNIX and NT.

Chapter 11, The PMDF Multithreaded Service Dispatcher, describes the PMDF Service Dispatcher which is used to oversee the handling of multithreaded services such as the POP3, IMAP, and SMTP services.

Chapter 12, The PMDF HTTP Server, describes some miscellaneous Dispatcher services, including the PMDF HTTP server which is used to serve out PMDF documentation and monitoring information.

Chapter 13, POP and IMAP Mailbox Servers, documents the mailbox servers (POP and IMAP servers) supplied with PMDF.

Chapter 14, Connection Authentication, SASL, and Password Management, describes connection authentication and password management.

Chapter 15, PMDF-TLS: Transport Layer Security, describes the optional layered product PMDF-TLS. In this chapter configuration and usage instructions are given for PMDF-TLS. PMDF-TLS provides for using Transport Layer Security to provide data encryption and integrity checking.

Chapter 16, Mail Filtering and Access Control, describes filtering of unwanted e-mail.

Chapter 17, The UNIX Local Channel, describes the local channel on UNIX systems, the single most important PMDF channel which is used by all PMDF configurations on UNIX. The UNIX Edition of the *PMDF User's Guide* provides documentation suitable for end users.

<REFERENCE>(CHAPTER_NTLOCAL\FULL), gives an overview of the local channel (normally the msgstore channel) on NT systems, a channel which is used by all PMDF configurations on NT. Complete information on the msgstore channel may be found in the *PMDF popstore & MessageStore Manager's Guide*.

Chapter 18, The Local, DECnet MAIL, and General MAIL_ Channels (OpenVMS), describes the local channel on OpenVMS systems, the single most important PMDF channel which is used by all PMDF configurations on OpenVMS. This chapter also describes MAIL-11 over DECnet (called DECnet MAIL in this document) channels.

Chapter 19, The PMDF User Interface on OpenVMS, decribes the PMDF interface presented to OpenVMS users. The OpenVMS Edition of the *PMDF User's Guide* provides documentation suitable for OpenVMS end users.

<REFERENCE>(CHAPTER_FORMS\FULL), describes the pop-up addressing forms which may be used to address FAXes, and query LDAP/X.500 and CCSO/ph/qi directory databases from within VMS MAIL, PMDF MAIL, and DECwindows MAIL.

Chapters 20–26 describe how to configure various additional PMDF channels.

Chapter 20, DECnet Channels (OpenVMS and Tru64 UNIX), describes two PMDF channels that run over DECnet transport:

- SMTP over task-to-task DECnet (OpenVMS only), and
- PhoneNet over task-to-task DECnet (OpenVMS only).

Chapter 21, TCP/IP Channels, describes one of the most important sorts of PMDF channel, SMTP over TCP/IP channels.

Chapter 22, Message Manipulation Channels, describes the Script and Conversion channels, which may be used to perform arbitrary manipulations on messages, such as virus and spam filtering.

Chapter 23, BSMTP Channels: MTA to MTA Tunnelling, describes Batch SMTP channels that can be used for MTA to MTA tunnelling.

Chapter 24, PhoneNet Channels (OpenVMS and UNIX), describes PhoneNet channels using the PhoneNet protocol over asynchronous terminal lines or modems.

Chapter 25, UUCP Channels (OpenVMS and UNIX), describes UUCP channels.

Chapter 26, Other Channels, describes additional channels, including:

- Alphanumeric pagers,
- Paper mail (e-mail directed to a printer),
- SMTP over an arbitrary I/O channels, and
- DEC Notes (OpenVMS only).

Chapter 27, The PMDF Queue to E-mail Symbiont (OpenVMS), describes the PMDF Queue to e-mail Symbiont which can be used to allow users of word processors to send FAXes directly from their word processing applications.

Chapter 28, E-mail Firewalls and Other E-mail Security Considerations, describes configuring PMDF for use as an e-mail firewall.

Chapter 29, Utilities on OpenVMS, documents the various PMDF utility programs available on OpenVMS platforms.

Chapter 30, Utilities on UNIX, documents the various PMDF utility programs available on UNIX and NT platforms.

Chapter 31, Monitoring, describes the PMDF counters and monitoring PMDF with DEC PolyCenter MAILbus Monitor, or with SNMP clients, or with web clients via the PMDF HTTP CGI.

Chapter 32, Performance Tuning, provides some ideas and hints on how to get the most performance out of PMDF.

Chapter 33, Maintenance and Troubleshooting on OpenVMS, provides some general guide lines for diagnosing problems with PMDF on OpenVMS and documents some of the more common problems that arise from time to time.

Chapter 34, Maintenance and Troubleshooting on UNIX, provides some general guide lines for diagnosing problems with PMDF on UNIX and documents some of the more common problems that arise from time to time.

Suggested starting points in this manual

PMDF includes a spectrum of features; the precise features of interest will vary greatly from site to site. A first reading of this manual might focus on a careful reading of Chapter 1 and Chapter 32, and then skimming Section 2.2 (omitting Section 2.2.6 on first reading), Section 2.3.1, Section 2.3.2, Section 2.4, Section 3.1, the beginning of Chapter 5, Chapter 8, the beginning of Chapter 9 (for OpenVMS sites) or Chapter 10 (for UNIX sites), the beginning of Chapter 11, if using POP or IMAP then Chapter 13, any channels discussed in Chapter 17 through Chapter 26 which you will be using (with particular attention to Chapter 21 discussing TCP/IP channels which are one of the most important sorts of channels for most sites), and Section 31.1.

Mail user agents

This manual focuses on PMDF's function as a Message Transfer Agent (MTA), to provide a uniform message distribution network that can be interfaced to multiple user interfaces (Mail User Agents, or MUA's). For further information on user interfaces, see documentation for that user agent, or the appropriate edition of the *PMDF User's Guide*. For instance:

VMS

On OpenVMS systems, PMDF uses the standard VMS MAIL facility as its primary user interface. PMDF also supplies a VMS MAIL-compatible user interface of its own, PMDF MAIL. PMDF MAIL is an extension of VMS MAIL which better understands network messaging (*e.g.,* supports RFC 822 and MIME) and uses the same message store as VMS MAIL. PMDF Pine, a port of the popular UNIX mail user agent Pine to OpenVMS, is also supplied as part of PMDF for OpenVMS and also uses the same message store as VMS MAIL. Information on PMDF MAIL and the OpenVMS-specific implementation details of PMDF Pine may be found in the OpenVMS Edition of the *PMDF User's Guide*. PMDF also supports Gold-Mail.

UNIX

On UNIX systems, PMDF can use as its mail user interface any such interface which normally submits its messages using sendmail or SMTP. For convenience, the PMDF distribution includes a copy of one such mail user interface for UNIX, the University of Washington's Pine.

Availability

PMDF software products are marketed directly to end users in North America, and either directly or through distributors in other parts of the world depending upon the location of the end user. Contact Process Software for ordering information, to include referral to an authorized distributor where applicable:

Process Software, LLC
959 Concord Street
Framingham, MA 01701 USA
+1 508 879 6994
+1 508 879 0042 (FAX)
sales@process.com

Volume I

The *PMDF System Manager's Guide* is in four volumes. Volume I comprises Chapter 1 through Chapter 13. Volume II comprises Chapter 14 through Chapter 28. Volume III comprises Chapter 29 through Chapter 34.

1 Structure and Overview

PMDF is a general-purpose, store-and-forward system for distributing computer-based mail. The term *store-and-forward* means that PMDF automatically handles the requeuing and retransmission of mail messages necessitated when network links or services are temporarily unavailable. In contrast to *mail user agents* (MUAs) such as VMS MAIL or Pine which are used to create and read electronic mail messages, PMDF is a *mail transport agent* (MTA) responsible for directing messages to the appropriate network transport and ensuring reliable delivery over that transport.

PMDF provides a uniform distribution environment that can be interfaced to multiple user interfaces (MUAs), networks, protocols, and transport mechanisms. As this interfacing, from the user's point of view, is accomplished transparently, PMDF presents to the user a homogeneous mail network; *i.e.,* PMDF seamlessly blends heterogeneous mail networks into a single, coherent mail system.

This chapter presents a very brief sketch of the internal structure and operation of PMDF, including an overview of the PMDF configuration file. The subsequent two chapters describe in detail this configuration file, the heart of PMDF: the domain rewriting rules used to rewrite addresses and route mail, and the channel/host table which establishes a site's available channels (message paths and gateways) and their characteristics. Subsequent chapters provide additional details on other facets of PMDF.

1.1 The Structure of PMDF

An analogy

One analogy for PMDF's function as an MTA (Mail Transport Agent) is that PMDF performs a similar function for e-mail messages that a central transportation transfer station performs in a city. In a city, passengers can come in to a station by way of any one of numerous possible transports— by foot, by road, by subway, by railroad, by air, *etc.* Over some of these transports there can be alternate protocol possibilities: for instance, taxi cabs, busses, and private cars can all travel over roads; or another example is that commercial airlines and private airplanes provide variant forms of air travel. Depending on each passenger's destination, the passenger departs by way of an appropriate transport and protocol to get him to his next destination.

In the context of electronic mail, protocols are generally a high-level (not necessarily network specific) language spoken between two mailers, whereas transports are the low-level, network specific details used to implement a protocol on a given network.

Thus e-mail messages can come in to PMDF by any one of a variety of transports and protocols—submitted directly by a local user, via TCP/IP as an SMTP message from an Internet system, via a dial-up modem using the PhoneNet protocol, via DECnet as a MAIL-11 message, via DECnet as an SMTP message, via UUCP, via an X.400

transport, via SNA, *etc.* PMDF then routes the message out using a transport and protocol appropriate for the message's destination address.

Note that PMDF not only serves as a mechanism for sending and receiving mail, but also serves as a centralized switching yard or "gateway" for routing and rerouting mail traffic between multiple network transports. The use of PMDF as a mail gateway allows a PMDF host to provide electronic mail access through its network facilities for other, less capable machines.

The transportation methods—PMDF channels

PMDF uses what are called *channels* to implement specific combinations of transports and protocols. Each different transport and protocol combination has a corresponding different PMDF channel. The PMDF postmaster initially configures PMDF telling PMDF what sorts of transports and protocols are in use at his site, and what sorts of destination addresses should be routed through which sorts of channels. For instance, at sites with an Internet connection, Internet addresses are normally routed through an SMTP over TCP/IP channel; but at sites with only a UUCP connection, Internet addresses would instead be routed through a UUCP channel. Once PMDF is so configured, PMDF handles message routing and delivery automatically—ordinary users need never be aware of this underlying transport and routing: they simply address and send their messages and PMDF automatically routes and delivers them appropriately.

The layout of transportation routes—the PMDF configuration file

PMDF's main configuration file, the `pmdf.cnf` file, contains the fundamental PMDF configuration information for a site, the information about what sorts of transports and protocols are in use (PMDF channel definitions), and the information about which destination addresses should be routed through which channels (PMDF rewrite rules). This PMDF configuration file will be discussed further later in this chapter in Section 1.2, as well as in detail in Section 2.2 and Section 2.3.

The PMDF configuration file thus contains a site's "layout" in terms of transports and protocols and which destinations are reachable via what transports: akin to the physical layout of railroad tracks, bus lines, airline routes, *etc.*, for a transportation transfer station in a city.

Arrivals trigger activity—PMDF channels run on demand

There is also the issue of scheduling: when do messages arrive and when are they delivered (when do the trains, buses, airplanes, *etc.*, arrive and depart)?

For incoming messages, in most cases the underlying transport is simply configured to hand the messages over to PMDF immediately whenever they come in: PMDF is "passive" or "data driven", waiting for the messages, which can come in at any time.[1]

[1] An example of PMDF's data driven nature is the case of local users on the PMDF system submitting messages from their own Mail User Agents. When a local user on the PMDF system sends a message, their Mail User Agent invokes PMDF images so that the Mail User Agent can hand the message over to PMDF.

For some protocol and transport combinations, the component of PMDF that awaits incoming messages for that protocol and transport combination is implemented as a *server*; for instance, to listen for and receive incoming SMTP over TCP/IP messages, PMDF includes an *SMTP server*. In other cases, for some transports where the transport itself is not able to actively hand the messages over to PMDF, PMDF is configured to periodically "poll" the sending side and ask for incoming messages.

As mentioned, PMDF is data driven. When an external source submits a message into PMDF, the receiving PMDF channel processes the message to check whither the message is destined and then typically immediately hands the message over to the appropriate outgoing channel and triggers the outgoing channel to itself run in turn. (In the case of a message with multiple recipients, the receiving channel hands the message over to multiple outgoing channels and triggers each of the outgoing channels to run.) In particular, note that for outgoing messages, by default PMDF normally always makes an immediate attempt to deliver each message.

To recapitulate, somewhat unlike the physical transportation transfer station analogy above, in PMDF the normal scheduling of outgoing message deliveries is "on-demand" and immediate—it is rather as if each new passenger could get their own railcar or airplane with immediate departure.

Delivering the passengers—the execution of PMDF channel processing jobs

The execution of the incoming direction of a PMDF channel can occur in a variety of contexts: in a user's own process (as for messages submitted by local users on the PMDF system), as a PMDF server process (as for incoming SMTP messages), or as a PMDF channel job triggered in some other way. The receiving channel immediately triggers the execution of a subsequent channel job to perform the next step of delivery of the message, and these subsequent jobs triggered by PMDF itself run in PMDF processing queues. On OpenVMS, regular OpenVMS Queue Manager queues are used for PMDF processing; thus normal OpenVMS queue handling can be used to control many details of PMDF processing. On UNIX and NT, PMDF processing queues are controlled by the PMDF Job Controller, which has its own configuration options. Configuration of exactly how and when channels run in processing queues can be used to control characteristics of PMDF operation.

Delivery retry attempts

PMDF is a store-and-forward message system. If PMDF's attempt to deliver a message encounters a temporary failure condition, then PMDF stores the message and later reattempts delivery. There are PMDF periodic jobs that run every so often re-attempting delivery of not yet delivered messages; see Section 1.4.3 later in this chapter. Eventually, if a message still cannot be delivered after repeated attempts, the PMDF periodic return job returns ("bounces") the message back to the sender. How long PMDF keeps trying to deliver messages is of course configurable—see Section 1.4.4.

The waiting room—the PMDF queue area on disk

While messages are awaiting processing and delivery by PMDF, they are stored as message files on disk in the PMDF queue area for the destination channel.

Summary

Thus there are five major aspects of PMDF operation:

1. *Layout*: The PMDF configuration describes how PMDF should handle messages, addresses, and message content.
2. *Receiving messages*: PMDF servers or incoming channel jobs receive messages. In general, PMDF is data driven. PMDF waits, listening for external agents to submit messages whenever they want.
3. *Processing outbound messages*: PMDF channel jobs running in PMDF processing queues deliver outbound messages. Note that a job in a PMDF processing queue can be either an immediate delivery job, or a periodic job re-attempting delivery of not-yet-delivered messages.
4. *Scheduling of delivery retries and polling*: For messages that do not get delivered upon first attempt, a periodic delivery job periodically reattempts delivery. Note that periodic jobs are: (a) a backup to the normal immediate delivery attempts of (3), or (b) something of an exception to the normal case of (2) wherein messages are submitted immediately by autonomous agents external to PMDF.
5. *Message storage*: The PMDF queue area is where message files are stored while being processed and awaiting delivery.

Additional facilities

In addition to the major issues of message routing and delivery discussed so far, sites often want to modify the addresses in e-mail messages, or convert message contents. Some simple address modifications are configured simply as part of the basic e-mail routing in the PMDF configuration file. For more complex needs, PMDF has extensive address handling facilities for address aliasing, directory lookups, mailing lists, centralized naming, *etc.*; see for instance Chapter 3. PMDF can also convert message contents from one character set to another, or convert message contents from various non-standard formats to MIME format; and in addition, PMDF has a facility for sites to hook in third party document convertors to perform desired attachment conversions, *e.g.*, from Wordperfect to Microsoft Word; see in particular Chapter 6.

1.2 The PMDF Configuration File: Channels and Rewrite Rules

The PMDF configuration file is the heart of the PMDF configuration. This main configuration file establishes the channels in use at a site and establishes which channels are responsible for which sorts of addresses via rewrite rules. Continuing the analogy of the previous section, the PMDF configuration file establishes the layout of the e-mail system by specifying the transport methods available (channels) and the transport routes (rewrite rules) associating types of addresses with appropriate channels.

Chapter 2 provides a detailed discussion of this file; an overview will be presented here.

Location

The PMDF configuration file is located via the logical PMDF_CONFIG_FILE (OpenVMS), the PMDF tailor file option PMDF_CONFIG_FILE (UNIX), or the Registry entry PMDF_CONFIG_FILE (NT). It is normally a file named pmdf.cnf, located in the PMDF table directory.[2]

Creation

The PMDF configuration file is normally initially created using the PMDF configuration utility, discussed in the *PMDF Installation Guide*. Alternatively, the file can be created entirely manually. Typically as site needs and configurations change, the PMDF postmaster will want to modify the PMDF configuration file, adding in new channels if new transport methods become available, fine tuning the operation of specific channels via numerous channel keyword optional settings, adding or changing rewrite rules as new systems are added or system names are changed.

Format overview

The PMDF configuration file is an ASCII text file that can be created or changed with a text editor. **The configuration file should be world readable. Failure to make the configuration file world readable can cause unexpected PMDF failures.**

The format of the configuration file consists of two parts: domain rewriting rules and the channel definitions. The domain rewriting rules appear first in the file and are separated from the channel definitions by a blank line. Each channel definition is separated from the following channel definition by a blank line.

Thus note that blank lines are significant in the PMDF configuration file: they delimit the end of the rewrite rules section of the file and separate channel definitions one from another.

Comment lines can appear anywhere in the configuration file. A comment is introduced with an exclamation point, !, in column one. Liberal use of comments to explain what is going on is strongly encouraged. Comment lines are ignored by the configuration file reading routines — they are essentially "not there" as far as the routines are concerned and do not count as blank lines.

[2] The PMDF table directory is the main PMDF directory for storing configuration files. The PMDF table directory is located via the logical PMDF_TABLE (OpenVMS), the PMDF tailor file option PMDF_TABLE (UNIX), or the Registry entry PMDF_TABLE (NT). Thus typically the PMDF configuration file is PMDF_TABLE:pmdf.cnf on OpenVMS, /pmdf/table/pmdf.cnf on UNIX, or on NT, has a full path such as C:\pmdf\table\pmdf.cnf, though the exact disk drive can vary depending upon installation.

The contents of other files can be included in the PMDF configuration file using the < PMDF file include operator. Channel and rewrite rule definitions for PMDF layered products, for instance, are commonly incorporated into the main PMDF configuration file by reading subsidiary files into the main configuration file.

Use and operation

When a message enters the PMDF system it must be placed into the proper channel queue or queues for transport or delivery out that channel. PMDF's message enqueue routines consult the PMDF configuration file to determine this: the recipient addresses in the incoming message are each individually processed through the PMDF rewrite rules to determine the proper channel (route) for each, and message copies are then written to those corresponding channel queue areas. The rewriting process is also capable of changing the address, for instance, converting local host nicknames (short-form names in RFC 822 terminology) into fully qualified Internet style domain names.

1.2.1 Channels

The central unifying construct in PMDF is the channel. A channel is some form of connection with another system or group of systems. Here, "system" is being used quite loosely and can mean another computer system, mail system, user agent, gateway, *etc*. The actual hardware connection or software transport or both can vary widely from one channel to the next. Only the PMDF manager need know anything about PMDF's channels; users are never aware of the existence of channels and only see a single, uniform interface regardless of how messages reach their destination.

Each channel consists of one or more channel programs and an outgoing message queue for storing messages that are destined to be sent to the systems associated with the channel. Channel programs perform two functions: (1) they transmit messages to other systems, deleting them from their queue after they are sent, and (2) they accept messages from other systems, placing them (*i.e.,* enqueuing them) into channel queues. Note that while a channel program only removes messages from its own queue it can enqueue messages to any queue whatsoever, including its own.

A channel program which initiates a transfer to another system on its own is called a *master* program, while a program which accepts transfers initiated by another system is called a *slave* program. A channel can be served by a master program, a slave program, or both. Either type of program can or can not be bidirectional; the direction in which a message is travelling can have nothing to do with the type of program that handles it. For example, in the case of a PhoneNet channel, the master and slave programs are both capable of transmitting and receiving messages.

Channel definitions appear in the lower section of the PMDF configuration file (following the first blank line in the configuration file). The channel definitions are collectively referred to as the *channel*/*host table*; an individual channel definition forms a *channel block*. There are a great many optional controls for channel behavior. See Section 2.3 for complete information on channel definitions.

1.2.2 Domain Rewriting Rules

Domain rewriting rules, or, as they are frequently called, *rewrite rules*, play two important roles. First, they are used to rewrite addresses into their proper or desired form. Secondly, they are used to determine to which channels a message should be enqueued.[3] The determination of to which channels a message should be enqueued is made by rewriting its envelope To: addresses.

Each rewrite rule appears on a single line in the upper half of the PMDF configuration file. Comments but not blank lines can appear between rules. Every rule consists of two parts: a pattern followed by an equivalence string or template. The two parts must be separated by one or more spaces. Spaces are not allowed in the parts themselves. The template specifies a mailbox name (*e.g.*, username), a host/domain specification, and the name of a system attached to an existing PMDF channel to which messages to this address should be enqueued.

See Section 2.2 for complete information on the use of rewrite rules.

1.3 Enabling PMDF to Receive Messages

The details of configuring transports to hand messages over to PMDF vary greatly from transport to transport, and it is not possible to provide much in the way of a general overview; specifics will be described in subsequent chapters discussing specific sorts of PMDF channels.

One item in particular can be pointed out, however, and that is that the PMDF Dispatcher, described in Chapter 11, controls the operation of PMDF server processes, such as the PMDF SMTP server (one of the most important PMDF components at a typical site).

1.4 Processing Jobs

PMDF uses processing jobs to run all master channel programs. On OpenVMS systems, these jobs take the form of either batch jobs or execution jobs depending upon whether a batch queue or Process Symbiont queue is used for the queue to which the processing jobs are submitted; unless you have configured PMDF otherwise, the use of batch queues is the norm. On UNIX and NT systems, processing jobs are managed by the PMDF Job Controller, discussed in detail in Chapter 10.

PMDF employs two kinds of processing jobs — periodic and immediate. Immediate jobs are used to process messages as they are submitted. Periodic jobs are jobs that run at fixed time intervals and process deferred requests of one sort or another. Periodic jobs resubmit themselves; immediate jobs do not. Periodic jobs typically handle the messages which immediate jobs were unable to deliver. On OpenVMS systems, periodic jobs, once initially submitted to an OpenVMS batch queue, reschedule (resubmit) themselves as needed. On UNIX systems, periodic jobs are scheduled by the `cron` daemon. On NT

[3] And thus the terminology implied by the often asked question, "To which channel does the message rewrite?"

systems, periodic jobs are scheduled by the Task Scheduler at the times specified with the `at` command.

1.4.1 Immediate Message Submission Jobs

Each time a message is placed in a channel queue for a channel that is marked as `master` and `immediate`,[4] PMDF attempts to start a processing job to deliver the message. The job is placed in the queue specified by the `queue` keyword in the channel definition. If no queue is specified, the default queue is used.

VMS

On OpenVMS systems, the default queue is MAIL$BATCH. PMDF requires CMKRNL and SYSPRV privileges in order to submit jobs. Note that PMDF does not use the $CMKRNL system service directly in submitting jobs: the OpenVMS services PMDF calls to submit jobs require that the calling process have the CMKRNL privilege.

On UNIX and NT, the PMDF Job Controller implements PMDF's job queuing system. The default queue is the queue named `DEFAULT` in the Job Controller configuration file.

PMDF always checks to see if a processing job is already pending to process messages on the channel requesting service. If such a job is in fact pending, then PMDF does not bother to create an additional, superfluous job. This strategy prevents large numbers of incoming messages from creating inordinate numbers of superfluous delivery processing jobs.

With this scheme, messages typically get delivered very quickly. Unfortunately, a price must be paid in terms of overhead — approximately one processing job is generated per message. If this overhead is unacceptable (either in terms of CPU overhead or in terms of the expense of making a connection), PMDF can, on a per-channel basis, be prevented from generating jobs on demand. See the discussions of the `periodic` and `after` channel keywords in Section 2.3 for details.

1.4.2 Manually Starting an Immediate Message Submission Job

It is sometimes useful to be able to start message delivery operations manually. For example, suppose that your Internet connection was down and while it was down a lot of messages built up in the outbound TCP/IP queues. The network is now up and you want to begin delivery now rather than wait for the periodic delivery job. The obvious thing to do next is to start a delivery job to deliver all the pending messages. One way to do this is to simply run `master.com` interactively from a suitably privileged account on an OpenVMS system, or to run the `pmdf run` utility from the `root` account on a UNIX system or from the `Administrator` account on an NT system; *i.e.*, on OpenVMS,

```
$ @PMDF_COM:master channel [polling-flag [since-time]]
```

or on UNIX,

[4] These are the defaults.

```
# pmdf run channel [polling-flag]
```

or on NT,

```
C:\> pmdf run channel [polling-flag]
```

Here *channel* is the channel to process and *polling-flag* is `poll` if the connection is to be established regardless of whether or not messages are queued for delivery. If *polling-flag* is `nopoll`, the default, a connection is made only if messages are queued for delivery. *since-time* is an optional date and time specification. Queue entries created before *since-time* will not be processed. Omitting *since-time* causes all queue entries to be processed.

The problem with this technique is that it ties up your terminal for the duration of the transaction. The alternative is to use the `submit_master.com` procedure on OpenVMS or the `pmdf submit_master` utility (or the synonymous `pmdf submit` utility) on UNIX or NT to submit a processing job that does the same thing. On OpenVMS, use a command of the form, (where *queue-name* will default to MAIL$BATCH if it is not specified):

```
$ @PMDF_COM:submit_master channel [polling-flag [queue-name [since-time]]]
```

or on UNIX, a command of the form:

```
# pmdf submit_master channel [polling-flag]
```

or on NT, a command of the form:

```
C:\> pmdf submit_master channel [polling-flag]
```

All the defaults are the same as when `master.com` (on OpenVMS) or `pmdf run` (on UNIX or NT) is invoked directly.

1.4.3 The Periodic Message Delivery Retry Job

A periodic job is one which reschedules itself for execution each time it runs. This section will discuss the first of PMDF's two main periodic jobs, the *periodic delivery job*. This job's primary purpose is to reattempt delivery of messages not yet delivered. (Note that in normal PMDF configurations, normal messages get an immediate delivery attempt by an immediate job, as described above in Section 1.4.1. Thus the periodic delivery job is primarily a delivery retry job—it is not the main mechanism for initial message delivery attempts.) The periodic delivery job is embodied on OpenVMS by the command procedure `PMDF_COM:post.com`, which runs every four hours by default (this value can be changed easily by setting the `PMDF_POST_INTERVAL` logical), or is embodied on UNIX by the shell script `/pmdf/bin/post.sh`, which the `cron` daemon is normally scheduled to run every four hours, or is embodied on NT by the program `post_job.exe` located in the PMDF binary image directory (normally `C:\pmdf\bin\`) which the `at` command normally schedules to run every four hours under the Task Scheduler.

The post job, whether embodied by the `post.com` command procedure or the `post.sh` shell script or the `post_job.exe` program, runs the `post` program which scans through all the channels currently defined in the configuration file. It also checks the corresponding queues for messages. Processing jobs are unconditionally submitted to run the master channel programs for any channels marked with the keyword `master` so as to poll remote systems that cannot establish their own connections. Jobs are also submitted for channels that support master channel programs and have messages queued. After this is done the post job then terminates. It will run again in another four hours.

The jobs `post` creates run in the queue appropriate to the channel (specified with the `queue` channel keyword); this can be a queue other than the one in which `post` itself runs.

1.4.3.1 Adjusting Periodic Delivery Retry Job Frequency

PMDF's suggested default behavior of running the periodic delivery job once every four hours is appropriate for most sites. Indeed, at busy sites, running the periodic delivery job too frequently tends to be counterproductive. Even if particular channels need to run more frequently, perhaps due to needing to poll to check for new incoming messages (*e.g.*, LAN channels), it is often best to leave the regular PMDF post job running at its usual frequency and to instead set up a special batch job that runs more frequently for the special channels; this is, in fact, the role played by the `pc_post` job for PMDF-LAN channels.

However, if a site does have a special need to run the periodic job more frequently, consider the following.

First, note that RFC 1123, Requirements for Internet Hosts, requires that Internet mail wait at least 30 minutes before being retried. *Do not run your channel to the Internet more frequently than every half hour.*

Next, if you must set PMDF_POST_INTERVAL to some small interval (OpenVMS) or have `cron` running the periodic jobs at some small interval (UNIX), or have the Task Scheduler running the periodic job at some small interval (NT), consider using a

```
defaults period n
```

channel, with *n* a suitable integer, to set the default channel periodicity back to something more like the usual four hour period, and mark only the channels that *need* to run more frequently with `period 1` so that only they run every time the periodic post job runs.

Finally, PMDF normally performs some periodic clean up tasks when the periodic delivery job runs. PMDF's defaults are tuned for the case where the periodic job only runs every couple of hours. If you will be running the periodic job more frequently, you should adjust PMDF's clean up task frequency, so that clean up tasks are not being executed needlessly often; see Section 1.4.3.2 below.

1.4.3.2 Clean Up Tasks Performed by the Periodic Delivery Job

The periodic delivery job normally performs some clean up tasks when it runs, such as purging back old versions of log files and every so often re-synchronizing the PMDF queue cache database.

By default, old log file versions are purged every time the periodic delivery job runs. The frequency at which this purging is performed can be controlled via the PMDF_VERSION_LIMIT_PERIOD logical (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT). The number of log file versions retained is controlled by the PMDF_VERSION_LIMIT logical (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT) and defaults to 5, if not specified.

By default, the PMDF queue cache database is re-synchronized every couple of times the periodic delivery job runs. The frequency of this re-synchronization can be controlled via the PMDF_SYNCH_CACHE_PERIOD logical (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT).

Note: On OpenVMS, the PMDF_SYNCH_CACHE_PERIOD and PMDF_VERSION_LIMIT_PERIOD logicals should not be used if MAIL$BATCH runs on more than one node in a cluster, as this can lead to unpredictable results.

1.4.4 Returning Undeliverable Messages

A periodic job is one which reschedules itself for execution each time it runs. This section will discuss the second of PMDF's two main periodic jobs, the *periodic return job*. This job is embodied on OpenVMS by the command procedure PMDF_COM:return.com, which runs once a day at 0:30:00 by default, or is embodied on UNIX by the shell script /pmdf/bin/return.sh, which the cron daemon is normally scheduled to run once a day at 30 minutes after midnight, or is embodied on NT by the program return_job.exe in the PMDF binary image directory (usually C:\pmdf\bin\) which the at command normally schedules to run once a day at 30 minutes after midnight under the Task Scheduler. This job is primarily used to return (bounce) old, undeliverable messages which have sat around in the message queues for too long. The frequency with which the PMDF return job runs can be altered, if desired; see Section 1.4.4.1 below.

The return job (after first, on OpenVMS, resubmitting itself to run again, at its next scheduled time, in the queue in which it is presently running) by default scans the channels listed in the configuration file each time it runs, checking the values established with the notices keyword. If any of the urgentnotices, normalnotices, and nonurgentnotices channel keywords have been used to set more specific timeouts for certain priorities of messages, the return job checks those values. The messages queued to each channel are then checked. A warning message is sent for every message whose age in days matches any of the values specified with the notices keyword on the associated channel—or in the case of the *notices keywords, any message whose priority matches or exceeds that specified by the keyword and whose age in days matches any of the keyword's values. The default ages if no notices keyword is specified are 3, 6, 9, and 12 days. If the message is as old or older than the final notices value, the entire message is returned and the original message is deleted from the channel queue; no further attempts to deliver the message will be made. The frequency with which the periodic return job attempts to perform this task of returning old messages can be

controlled via the PMDF_RETURN_PERIOD logical (OpenVMS) or PMDF tailor file option (UNIX) or NT Registry entry.

Note: On OpenVMS, the PMDF_RETURN_PERIOD logical should not be used if MAIL$BATCH runs on more than one node in a cluster, as this can lead to unpredictable results.

The text of the warning and failure notices issued by the message return system is contained in the return_*.txt files located in the PMDF language-specific directory.[5] These files can be edited to provide different notification text if desired.[6]

PMDF maintains a history of delivery attempts for each message, which sometimes will include information about failed delivery attempts. This information is included in returned messages if RETURN_DELIVERY_HISTORY is set to 1 in the PMDF option file (this is the default). A value of 0 disables the inclusion of this information.

Finally, the message return subsystem normally performs some clean up tasks in addition to returning old messages; these additional functions are described below in Section 1.4.4.2.

1.4.4.1 Adjusting Return Job Frequency

VMS

On OpenVMS, if RETURN_UNITS=1 is specified in the PMDF option file, then the return job will run every hour instead of once a day.

UNIX

On UNIX systems, the frequency of the PMDF return job is controlled via the crontab entry for /pmdf/bin/return.sh. If the return job is scheduled to run more frequently than once a day, as for instance on an hourly basis, then RETURN_UNITS=1 should be set in the PMDF option file, so that notices values will be interpreted in hours, rather than in days as is the default.

On NT systems, the frequency of the PMDF return job is controlled via the at command which sets an entry for running return_job.exe under the Scheduler. If the return job is scheduled to run more frequently than once a day, as for instance on an hourly basis, then RETURN_UNITS=1 should be set in the PMDF option file, so that notices values will be interpreted in hours, rather than in days as is the default.

[5] On UNIX systems, this is the directory specified by the PMDF_LANG setting in the /etc/pmdf_tailor file; on NT systems, this is the directory specified by the PMDF_LANG entry in the PMDF tailor Registry; on OpenVMS systems, this is the directory specified by the PMDF_LANG logical. Usually the PMDF language-specific directory is simply a synonym for the PMDF table directory.

[6] As the text of such a file is copied into messages, certain substitutions are made. A %C expands into the number of days the message has been queued; %L expands into the number of days the message has left in the queue before it is returned; %F expands into the number of days a message is allowed to stay in the queue; %S [%s] expands into an S [s] if the previously expanded numeric value was not equal to one; %U [%u] expands into the units, Hour [hour] or Day [day], in use; %R expands into the list of the message's recipients; %H expands into the message's headers.

If the PMDF return job is running once an hour, then the default will be to issue warning notices for messages which have remained undeliverable for 3, 6, or 9 hours. Messages which have remained undeliverable for 12 or more hours are returned in their entirety to their sender and no further delivery attempts are made. *Note: When RETURN_UNITS=1, these defaults will result in mail being bounced much too soon; therefore, sites are strongly encouraged to use the* `notices` *channel keyword to set "bounce" ages in excess of twelve hours.*

As the PMDF return job also performs various PMDF periodic cleanup tasks, tuned on the assumption that the return job will only be running once a day, when the PMDF return job is run more frequently various PMDF parameters should be adjusted accordingly. In particular, the `PMDF_RETURN_SYNCH_PERIOD`, `PMDF_RETURN_SPLIT_PERIOD`, and `PMDF_RETURN_CHECK_PERIOD` logicals (OpenVMS) or `PMDF_RETURN_SYNCH_PERIOD` and `PMDF_RETURN_SPLIT_PERIOD` PMDF tailor file options (UNIX) or `PMDF_RETURN_SYNCH_PERIOD` and `PMDF_RETURN_SPLIT_PERIOD` Registry entries (NT) should generally be adjusted so that these tasks are still performed only once a day; see Section 1.4.4.2.

1.4.4.2 Clean Up Tasks Performed by the Return Job

The periodic return job normally performs various clean up tasks when it runs, such as rolling over the `mail.log*` files, and if separate connection logging is being used then rolling over the `connection.log*` files also, re-synchronizing the PMDF queue cache database, and purging old log files and (on OpenVMS) resetting log file version numbers.

By default, the periodic return job checks each time it runs for any `mail.log*` or `connection.log*` files in the PMDF log area. It appends any existing `mail.log_yesterday` file to the cumulative log file, `mail.log`, renames the current `mail.log_current` file to `mail.log_yesterday`, and then begins a new `mail.log_current` file. The return job also performs the analogous operations for `connection.log*` files. The frequency at which the periodic job performs these log file roll overs can be controlled via the `PMDF_RETURN_SPLIT_PERIOD` logical (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT).

The return job by default also re-synchronizes the PMDF queue cache database each time it runs, scanning all the messages in the queues and entering any missing messages into the PMDF queue cache. The frequency with which the return job performs queue cache database re-synchronization can be controlled via the `PMDF_RETURN_SYNCH_PERIOD` logical (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT).

VMS

On OpenVMS, the queue cache is also periodically subjected to either a CONVERT operation, or to a CONVERT/RECLAIM operation to reclaim any unused space.

VMS

On OpenVMS, the PMDF return job also resets the version numbers of all log files in the PMDF log directory, `PMDF_LOG:` on OpenVMS. Unfortunately, the version numbers on open files cannot always be changed. Therefore, if, after resetting the version numbers, any log files have a version number exceeding the warning level, 25,000, set in the command procedure `PMDF_COM:pmdf_check_logs.com`, then a mail message will be sent to the Postmaster. When such a message is received the Postmaster must manually delete or reset the version numbers on the log files. Failure to do so will cause the

associated channel to stop working should the version number of one of its log files attain 32,767. The frequency at which the periodic return job checks log file versions can be controlled via the PMDF_RETURN_CHECK_PERIOD logical; the default, if the logical is not defined, is to check each time the return job runs.

The logical names (OpenVMS) or PMDF tailor file options (UNIX) or Registry entries (NT) described above for controlling the frequency of return job tasks are not defined by default. To use such a logical name or tailor file option, define the logical or set the tailor file option or Registry entry to an integer value N; that will cause the associated action to only be performed every N times the periodic return job runs.

Note: On OpenVMS, the PMDF_RETURN_SYNCH_PERIOD, PMDF_RETURN_SPLIT_PERIOD, and PMDF_RETURN_CHECK_PERIOD logicals should not be used if MAIL$BATCH runs on more than one node in a cluster, as this can lead to unpredictable results.

The periodic return job also includes a hook for executing a site-supplied clean up procedure. OpenVMS sites can provide their own PMDF_COM:daily_cleanup.com DCL procecure; UNIX sites can provide their own /pmdf/bin/daily_cleanup shell procedure; NT sites can provide their own daily_cleanup command script in the PMDF binary image directory (usually C:\pmdf\bin\). The periodic return job will automatically execute such a procedure, if it exists.

1.4.5 Managing Processing Job Execution on OpenVMS

On OpenVMS, the functions of PMDF's processing jobs can optionally be performed by detached processes (managed by the PMDF Process Symbiont), rather than running in plain OpenVMS batch queues. This improves performance by reducing the amount of process creation overhead associated with plain batch queues. See Chapter 9 for details on how to set up a detached processing environment for PMDF.

Regardless of whether plain OpenVMS batch queues or PMDF Process Symbiont queues are used for PMDF processing, the design of running PMDF jobs in queues allows control of PMDF activity by stopping and starting the queue. The use of queue oriented processing jobs also makes it possible to use PMDF in complex cluster environments, where PMDF can need access to communications hardware or software that resides on a system other than the one where the message processing activity originated. Generic queues can be used to spread the load caused by mail delivery activities across multiple machines in a cluster.

All PMDF processing jobs run by default in the MAIL$BATCH batch queue. MAIL$BATCH can either be a batch queue in its own right or a logical name that translates to the name of a batch queue. Alternate queues can be selected for jobs on a channel by channel basis by using the queue channel keyword. If the Process Symbiont is used, then the MAIL$BATCH queue will be a server queue, feeding multiple Process Symbiont execution queues.

In an OpenVMS cluster environment, note that the MAIL$BATCH queue can be a generic queue feeding specific queues on one or more systems thereby spreading the processing load around. If you do spread the load around, be sure that each system has the necessary resources to handle the types of traffic submitted to the MAIL$BATCH

queue. Such restrictions are generally dealt with by creating separate queues for channels requiring software or other resources only available on a few systems and then selecting those queues with the `queue` channel keyword.

1.4.6 Running Processing Jobs Under a Username Other than SYSTEM on OpenVMS

On OpenVMS, all PMDF processing jobs normally execute under the username SYSTEM. In some environments it can be necessary to change this default. The default username for periodic processing jobs is established when PMDF is installed; the installation procedure prompts for the username to be used.

The username must be that of an account with full privileges, *i.e.*, a privilege mask of -1.

To change to a different username after installation, redefine the system-wide, executive-mode logical name `PMDF_BATCH_USERNAME` to translate to the desired username and the system-wide, executive-mode logical name `PMDF_BATCH_ACCOUNT` to translate to the corresponding accounting field. The next step is to edit the two files `SYS$STARTUP:pmdf_startup.com` and `PMDF_COM:pmdf_submit_jobs.com`, and change their definitions of these logical names too. Next kill the periodic PMDF jobs in MAIL$BATCH, in particular:

- the periodic delivery job, and
- the message bouncer job,

and any other periodic jobs that can be present, depending on just what components of PMDF you are using, such as:

- the popstore message bouncer job,
- the PC channel polling job, and
- the DECUS UUCP return job.

Such periodic jobs must be killed (since if left alone they would continue to resubmit themselves using the SYSTEM username) and then resubmitted afresh under the new username by executing `PMDF_COM:pmdf_submit_jobs.com`.

1.5 Storage of Message Files on Disk

Messages being processed by PMDF are stored as files on disk, in the PMDF queue area, as described in Section 1.5.1 and Section 1.5.2 below.

1.5.1 Channel Queue Formats

Messages queued for delivery are always stored in the same format, described below in Section 1.5.2, regardless of the type of channel in which they are queued. All messages are stored in subdirectories under the PMDF queue directory.[7] Each channel has its own subdirectory, with the name of the subdirectory being that of the channel. For example, messages queued for delivery to Internet destinations are typically handled by a tcp_local channel, hence stored in `PMDF_QUEUE:[tcp_local]` on OpenVMS systems, `/pmdf/queue/tcp_local/` on UNIX systems, and usually `C:\pmdf\queue\tcp_local\` on NT. A channel subdirectory can in turn have multiple subdirectories beneath it, usually purely for file system usage reasons,[8] but also some channels, such as the MAILSERV and PRINTER channels, use a subsidiary `spool` subdirectory for storing temporary file. Each file in a channel subdirectory contains a single message.

Note: On OpenVMS, some temporary files are stored in the top level queue directory. The names of these temporary files usually begin with a dollar sign, `$`.

All of these directories are protected against access by nonprivileged users. The first two characters of each file name are a representation of the number of times delivery has been attempted on the file. This information is encoded in "complemented base 36". Files are initially created having names beginning with "`ZZ`". Upon completion of an unsuccessful delivery attempt, a file is renamed, counting down in complemented base 36. So a name beginning with "`ZY`" indicates that at least one attempt has been made, a name beginning with "`ZA`" indicatees that at least 25 attempts have been made, a name beginning with "`Z0`" indicates that at least 35 attempts have been made, a name beginning with "`YZ`" indicates that at least 36 attempts have been made, and so on.

The remainder of the file names are pseudo-random strings of hexadecimal characters that serve to make the file names unique. The file type (extension) is always a pair of letters or digits, usually "`00`". Messages being held have a "`.HELD`" as the file type; these messages are not eligible for delivery processing and must be renamed to have a two-character file type before they will become eligible for delivery.[9]

The actual internal format of the message files is irrelevant. Those who want to write their own PMDF channel programs should access the messages via the documented PMDF API interface. This interface is documented in the *PMDF Programmer's Reference Manual*. PMDF itself uses the MM subroutine library to enqueue messages and the QU subroutine library to dequeue messages. However, it is sometimes useful for a system manager to examine messages in the queues, so it is nice to note that messages are stored as ASCII text and message files can be typed on a terminal without adverse effects. An overview is provided of message file format below in Section 1.5.2.

[7] `PMDF_QUEUE:` on OpenVMS; `/pmdf/queue/` on UNIX; the directory specified by the PMDF_QUEUE Registry entry on NT, usually `C:\pmdf\queue\`.

[8] See the `subdirs` channel keyword in Section 2.3.4.17.

[9] See Section 33.4.7 or Section 34.4.7 for information on dealing with files marked `.HELD`, should you ever encounter any.

1.5.2 Message File Structure

Most PMDF messages are stored as text files. Multimedia mail support for various sorts of binary information, such as OpenVMS binary files, compound documents, and image data are supported via printable text encodings. Messages with multiple parts (possibly containing different types of data) are represented as a series of text sections separated by special unique delimiter strings.

A sample mail message file is given in Figure 1–1 below.

Figure 1–1 Sample Mail Message File

```
u;MIRANDA
c;l
s;EXAMPLE.COM
i;01G6YTYFU6748WWH0Y@EXAMPLE.COM
h;<01G6YTYFU6748WWH0Y@EXAMPLE.COM>
m;MIRANDA@EXAMPLE.COM  ❶
j;rfc822
f;prospero@EXAMPLE.COM
prospero@ISLAND.EXAMPLE.COM
Boundary_(ID_bNmDRTvfQNkeUUBbOugFTQ) ❷
Received: from EXAMPLE.COM by EXAMPLE.COM (PMDF V6.1 #9441) ❸
 id <01G6YTYFU6748WWH0Y@EXAMPLE.COM> for prospero@EXAMPLE.COM;
 Fri, 15 Nov 2012 14:55:51 EDT
Date: Fri, 15 Nov 2012 14:55:48 -0500 (EDT)
From: "Miranda" <miranda@example.COM>
Subject: Woe the day
To: prospero@example.COM
Message-id: <01G6YTYFU6748WWH0Y@example.COM>
MIME-version: 1.0
Content-type: TEXT/PLAIN; CHARSET=US-ASCII
Content-transfer-encoding: 7BIT
X-Envelope-to: prospero@example.COM
X-VMS-To: IN%"prospero@example.com"
 ❹
Prospero,

   More to know
   Did never meddle with my thoughts.

                                Miranda
Boundary_(ID_bNmDRTvfQNkeUUBbOugFTQ) ❺
```

Briefly, the key items in each message file are

❶ The message envelope. The first records in the file contain message envelope (*i.e.,* transport) information.

❷ The envelope is terminated either by a line containing a boundary marker, or by a line containing two CTRL/A characters.

❸ The header lines of the message follow the envelope; their format is mandated by RFC 822.[a]

❹ There can be any number of message header lines; the message header formed by this collection of header lines is terminated by a single blank line after which follows the message body.

❺ The message is terminated by a boundary marker matching the boundary marker at the beginning of the message, (or by a sequence of five CTRL/As if the message start was indicated using CTRL/As).

Process Software reserves the right to change this format in future releases of PMDF. User-written applications that either read or write queued PMDF message files should make use of appropriate PMDF library routines; see the *PMDF Programmer's Reference Manual*. Use of the PMDF API will insulate user applications from any future message format changes.

1.6 Other Important Files

The PMDF configuration file, presented in Section 1.2 above and described in more detail in Chapter 2, is the single most important PMDF file. Of only slightly lesser import is the alias file, described in Chapter 3. The existence of these two files is required. Listed below are some additional PMDF files and databases which are described elsewhere in this documentation:

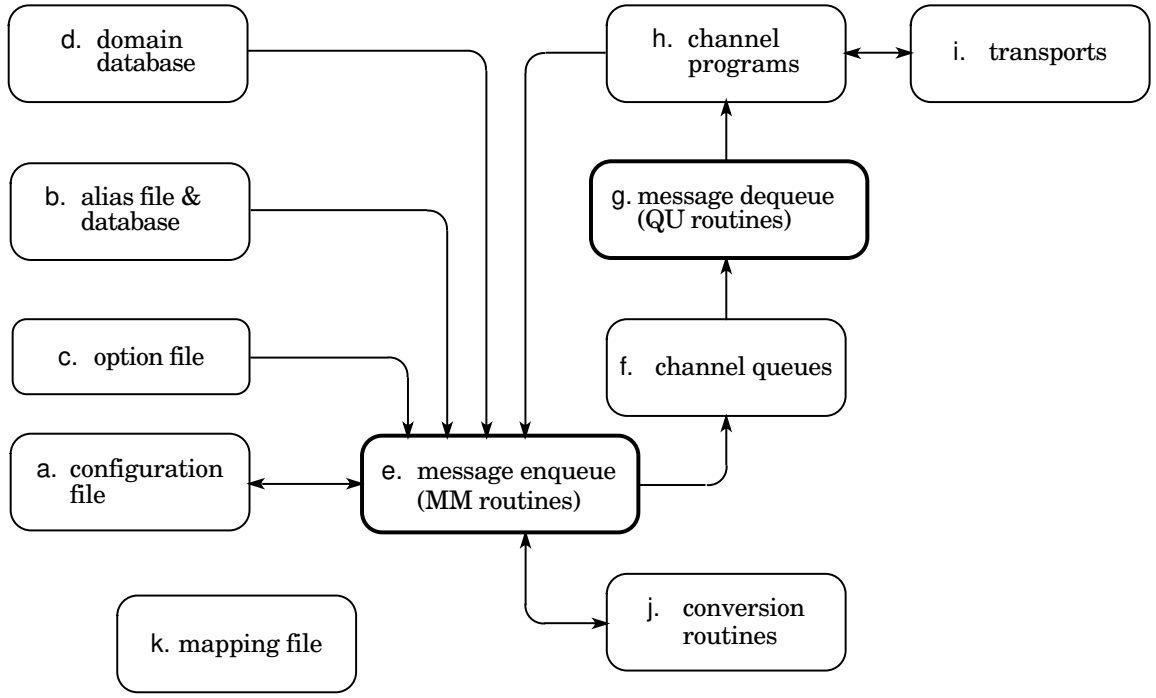
File or database	Described in	Usage
Alias file and database	Chapter 3	Implements aliases, mail forwarding, and mailing lists.
Character set file	Chapter 29 and Chapter 30	Character set tables; used to translate between character sets.
Channel option files		Many channels use channel option files to set options particular to the channel. See the relevant channel documentation for details on what options, if any, are available for the channel.
Compiled configuration file	Section 8.1	Shareable image built with PMDF's `cnbuild` utility; contains information from the alias, configuration, mapping, conversions, security, system wide filter file, and PMDF option files; speed up PMDF's performance through the use of a compiled configuration.

[a] A copy of RFC 822, *Standard for the Format of ARPA Internet Text Messages*, written by David Crocker can be found under the PMDF documentation directory, usually `pmdf_root:[doc.rfc]` on OpenVMS systems or `/pmdf/doc/rfc` on UNIX systems or the `rfc` subdirectory of the PMDF documention directory (usually `C:\pmdf\doc`) on NT.

File or database	Described in	Usage
Conversion file	Section 22.1	Used by the conversion channel to control message body part conversions; used to convert message attachments from one format to another.
Dispatcher configuration file	Section 11.3	Defines the PMDF server processes.
Domain database	Section 2.2.9	Domain rewriting rules stored in a database file; used primarily by BITNET configurations.
Forward database	Section 3.4	Address forwarding database, used primarily with autoregistered addresses.
General database	Section 2.2.6.5	Used for arbitrary, site-specific purposes.
IMAP server configuration file	Section 13.2.3.2.1	Sets options for the operation of the IMAP server
Job Controller configuration file	Section 10.1	On UNIX, defines the queues in which channels can run.
Log files	Section 2.3.4.84	There are two sorts of log files: those generated to record the run of a channel program (*e.g.,* a batch processing log file or channel debug log file), and the PMDF message and connection log file(s) such as `mail.log`, recording message flow through PMDF.
Mapping file	Chapter 5	Used by many different PMDF facilities as a repository of mapping tables; mapping tables are pattern based rules for transforming text-based data.
Message files	Section 1.5.1	Enqueued messages are stored in message files; message files are stored in channel queue directories.
Personal alias database	Section 3.1.3	User-level alias databases created with PMDF's `db` utility.
PMDF option file	Chapter 7	File of global PMDF options. Not to be confused with channel keywords specified in the configuration file or channel option files described in the channel documentation.
POP server configuration file	Section 13.2.3.2.2	Sets options for the operation of the POP server
Queue cache database	Chapter 29 and Chapter 30	The messages currently enqueued to PMDF and awaiting delivery are recorded in the queue cache database. Channel programs determine which messages to process by querying this database. PMDF's cache utility is used to manage this database.
Reverse database	Section 3.3.2	Database used to change addresses in outgoing mail messages; *e.g.,* change alonso@naples.example.com into King.Alonso@Example.Com.
Security configuration file	Section 14.2	File controlling PMDF's authentication source and SASL use.

A schematic of these PMDF files and components and how they interact is shown in Figure 1–2 below.

Figure 1–2 The Structure of PMDF



The various components shown in Figure 1–2 are as follows.

a. The configuration file defines the specific channels and thus the transports that a particular PMDF installation can use. The configuration file also contains additional domain definitions like those in the domain database, except those in the configuration file are in plain text. This file *must* be present and be readable by all users of PMDF; if it is missing or unreadable PMDF will abort with an initialization failure.

b. The alias file is used to establish aliases and mailing lists, a special case of an alias. The alias file *must* be present and readable by all users of PMDF. The alias database performs functions similar to the alias file, except that it is an indexed file containing a list of mailbox names on the local system that are actually aliases for different addresses or collections of addresses. The presence and use of this database is optional.

c. The option file is used to set various PMDF parameters which are not specific to any one channel. In particular, the sizes of various tables internal to PMDF can be controlled using entries in this file.

d. The optional domain database is an indexed file containing information about the various domains a particular PMDF installation can reach. This database tells PMDF how addresses for the various domains should be formatted and what transport should be used to deliver messages to or obtain messages from a given domain.

e. PMDF uses a common set of routines, the MM routines, to enqueue messages in the channel queues.

f. The channel queues are used to store messages prior to a channel program's sending them out via a transport.

g. PMDF uses a common set of routines, the QU routines, to dequeue messages from the channel queues.

h. Channel programs are the parts of PMDF that interface to the various transports.

i. PMDF uses various transports to actually send and receive messages. Some of these are part of PMDF (*e.g.,* PhoneNet) and some are not (*e.g.,* DECnet, TCP/IP).

j. The conversion routines implement the character set converters as well as the message encode/decode facilities used for handling encoded messages.

k. The mapping file is a repository for general mapping information used throughout PMDF. This file consists of a series of named tables written in a standard format. Various components of PMDF check for and use different tables for different purposes. For example, the SEND_ACCESS mapping table is used to control who can or can not send or receive mail from various channels.

1.7 Installation Environment: Logicals (OpenVMS), Tailor File (UNIX), Registry (NT)

When PMDF is installed, the installation procedure creates a file of PMDF logical name definitions (OpenVMS) `SYS$STARTUP:pmdf_startup.com`, or a PMDF tailor file of option settings (UNIX) `/etc/pmdf_tailor`, or PMDF entries in the NT Registry (see the Registry under `HKEY_LOCAL_MACHINE`, `SOFTWARE`, Process Software, PMDF, Tailor) establishing PMDF's operational parameters, such as specifying the normal names of PMDF's main configuration files and specifying on which disk(s) PMDF directories are located. For instance, the `PMDF_QUEUE` logical name (OpenVMS) or tailor file option (UNIX) or Registry entry (NT) specifies the location of the PMDF queue area.

On OpenVMS, a command to execute `pmdf_startup.com` should be added to the system startup procedure, so that these required PMDF logicals are properly defined each time the system is rebooted. On UNIX and NT, the tailor values are automatically consulted by PMDF as necessary.

Normally the Process Software-supplied `pmdf_startup.com` file, PMDF Tailor file, or NT Registry entries, once generated during PMDF installation, should not be site modified.[b] On OpenVMS, if you want to site customize such settings you can create a file `PMDF_COM:pmdf_site_startup.com` containing the settings that you want to alter; `pmdf_startup.com` will automatically execute such a site-supplied file, if it exists.

[b] The one value that is routinely changed is the OpenVMS-only logical name `PMDF_TIMEZONE`, which should be changed whenever the local timezone is changed. Values controlling various "clean up" task timing are also sometimes changed, as discussed above in Section 1.4.3.2 and Section 1.4.4.2.

Note: It is sometimes desired to change the values specifying the disk location of directories such as the PMDF queue, table and log directories. This should only be done when a system is running standalone (with no other processes on it). Furthermore, note that due to limitations in the Solaris utilities `pkginfo`, `pkgrm`, and `pkgadd`, manually relocating the entire PMDF directory tree (`/pmdf` and everything under it) after installation is not supported on Solaris.

1.8 Compliance with Standards

PMDF is fully compliant with RFC 822, and with RFCs 2045–2049 (MIME)[c], RFC 2183 (Content-disposition: header in MIME messages), RFCs 1892 and 1984 (Notification message format), and RFC 2298 (Message Disposition Notifications), the standards for the format of Internet text messages. SMTP support complies with RFC 821 (Simple Mail Transfer Protocol), and RFCs 1652, 1869, 1870, 1891, and 1985 (SMTP extensions), and RFC 2034 (Enhanced SMTP error return codes). PMDF's use of the Domain Name System for message routing complies with RFC 974. For blocking unsolicited bulk e-mail (spam), PMDF fully supports RFC 2505 (Anti-Spam Recommendations for SMTP MTAs). PMDF also complies with various other Internet formats and protocols, including RFC 1123 (Internet host application requirements), and RFC 976 (UUCP mail interchange).

PMDF's POP server is compliant with RFC 1939 (POP3).[d] PMDF's POP server also supports RFC 2449 (POP3 CAPA command). PMDF's IMAP server is compliant with RFC 2060 (IMAP4rev1).[e] PMDF's IMAP server also supports RFC 2342 (IMAP4 NAMESPACE command). PMDF's message store IMAP server also supports RFC 2086 (IMAP4 ACL extension), RFC 2087 (IMAP4 QUOTA extension), RFC 2088 (IMAP4 non-synchronizing literals), and RFC 2359 (IMAP4 UIDPLUS extension)

For user authentication during IMAP, POP, or SMTP connections, PMDF supports RFC 2222 (SASL; Simple Authentication Security Layer) and RFC 2554 (ESMTP AUTH).

The LDAP support in the PMDF directory channel supports RFC 2251 (LDAPv3), RFC 2252 (LDAPv3: Attribute Syntax Definitions), RFC 2253 (LDAPv3: UTF-8 DNs), RFC 2254 (LDAP Search Filters), and RFC 2255 (LDAP URL Format).

Regarding monitoring, PMDF supports RFC 1566 (Mail Monitoring MIB).

Copies of several of these and other standards can be found in the PMDF documentation directory; see `pmdf_root:[doc.rfc]` on OpenVMS or `/pmdf/doc/rfc/` on UNIX or the `rfc` subdirectory of the PMDF documentation directory (usually `C:\pmdf\doc\`) on NT.

Note that PMDF can perform all the functions necessary to administer mail within a domain, but does not itself provide domain name services. A name server is necessary to fully administer an Internet domain; such support is provided by a TCP/IP networking package and not by a mailer, since full domain support involves much more than handling mail messages. Thus, if PMDF is to be used in the Internet environment,

[c] RFCs 2045–2049 update RFCs 1521 and 1522, which originally defined MIME format.

[d] RFC 1939 updates RFC 1725, itself an update of RFC 1460 which updated RFC 1225.

[e] RFC 2060 is an update of RFC 1730 which originally defined IMAP4; IMAP4 itself updates IMAP2 (RFC 1176).

some facility should be used to perform domain name server functions for PMDF. This facility can reside on the system running PMDF, as part of the vendor-supplied TCP/IP implementation, or on another system on the network accessible to PMDF.

2 The Configuration File: Domain Rewrite Rules & the Channel/Host Table

The PMDF configuration file is the heart of the PMDF configuration. This file establishes, via channel definitions and domain rewriting rules, what types of connections the PMDF system has to other systems and mailers, what addresses are local to the PMDF system or local to the PMDF site, and how to route corresponding messages, optionally altering addresses in the process. The types of connections to other systems and mailers are established by means of the unifying PMDF construct, the channel. The altering of addresses and the routing of corresponding messages to channels are controlled by the domain rewriting rules.

Operationally, when a message enters the PMDF system it must be placed into the proper channel queue or queues. PMDF's message enqueue routines consult the PMDF configuration file to determine the proper channel queues. Each recipient address is processed through the domain rewriting rules to determine to which of the defined channels to enqueue the message. In addition, domain rewriting rules may also modify addresses; for instance, the message's addresses must be rewritten to eliminate any reference to local system nicknames or aliases (called short-form names in RFC 822) since these are not allowed in outgoing messages. All information about local short-form names and how to eliminate them via rewriting is usually contained in the configuration file or a subsidiary database such as the domain database discussed elsewhere.

On OpenVMS systems, the configuration file is pointed at by the logical PMDF_CONFIG_FILE;[1] on UNIX systems the configuration file name and location are specified by the PMDF_CONFIG_FILE[2] setting in the PMDF tailor file.

The PMDF configuration file is an ASCII text file that can be created or changed with a text editor. **The configuration file should be world readable. Failure to make the configuration file world readable may cause unexpected PMDF failures.**

Note: NT sites may find the `pmdf edit` configuration editor application especially convenient for editing PMDF configuration files.

2.1 Structure of the Configuration File

The configuration file consists of two parts: domain rewriting rules and the channel definitions. The domain rewriting rules appear first in the file and are separated from the channel definitions by a blank line; see Section 2.2 for details on domain rewriting rules. The channel definitions are collectively referred to as the *channel/host table*; an individual channel definition forms a *channel block*. See Section 2.3 for details on channel definitions. Examples of simple configuration files, illustrating the overall structure of the configuration file, may be found in Section 2.4.

[1] Usually `PMDF_TABLE:pmdf.cnf`.

[2] Usually `/pmdf/table/pmdf.cnf`.

2.1.1 Blank Lines in the Configuration File

Blank lines are significant in the configuration file. The first blank line in the file terminates the domain rewriting rules section of the file and marks the beginning of the channel definition section of the file. At least one blank line must also be present between each distinct channel definition.

2.1.2 Comments in the Configuration File

Comment lines may appear anywhere in the configuration file. A comment is introduced with an exclamation point, `!`, in column one. Liberal use of comments to explain what is going on is encouraged.

It is important to distinguish between blank lines and comment lines. Blank lines play an important role in delimiting sections of the configuration file. Comment lines are ignored by the configuration file reading routines — they are essentially "not there" as far as the routines are concerned and do not count as blank lines.

2.1.3 Continuation Lines in the Configuration File

Long lines may be continued by ending them with a backslash, `\`.

Note that even comment lines may be continued with a backslash, for instance:

```
! This is all \
 a single comment line
```

2.1.4 Including Other Files in the Configuration File

The contents of other files may be included in the configuration file. If a line is encountered with a less than sign, <, in column one, then the rest of the line is treated as a file name; the file name should always be a full file path. The file is opened and its contents are spliced into the configuration file at that point. Include files may be nested up to three levels deep.

Note: Any files included in the configuration file must be world readable just as the configuration file is world readable.

2.2 Domain Rewriting Rules

Domain rewriting rules are kept in the upper portion of the PMDF configuration file and, when there are many rules as in the case of a BITNET site, in an auxiliary database called the domain database. See Section 2.1 for an overview of the PMDF configuration file format.

Throughout much of this manual, domain rewriting rules are referred to as simply "rewrite rules".

2.2.1 The Purpose of Domain Rewriting Rules

Domain rewriting rules are used to convert addresses into true domain addresses and to determine their corresponding channels. These rules are used to rewrite addresses appearing in both the transport layer and the message header. The transport layer is the message's "envelope", which contains routing information and is invisible to the user.

The rewrite rules and the table of channels cooperate to determine the disposition of each address. The result of the rewrite process is a rewritten address and a "routing system"; *i.e.,* the system to which the message is to be sent. Depending upon the topology of the network, the routing system may only be the first step along the path the message takes to reach its destination or it may be the final destination system itself.

After the rewrite process has finished a search is made for the routing system among the channel portion of the configuration file. Each channel will have one or more host names associated with it. The routing system name is compared against each of these names to determine to which channel to enqueue the message.

Note that PMDF provides other means of manipulating addresses for the purposes of changing them. See for instance Chapter 3.

2.2.2 Location and Format of Domain Rewriting Rules

The rewrite rules appear in the upper half of the PMDF configuration file. On OpenVMS systems, this is the file pointed at by the logical PMDF_CONFIG_FILE;[4] on UNIX systems this is the file specified by the PMDF_CONFIG_FILE[5] setting in the PMDF tailor file, `/etc/pmdf_tailor`. Each rule in the configuration file appears on a single line. Comments but not blank lines are allowed between rules. The end of the rewrite rules is denoted by a blank line after which follow the channel definitions.

Every rule consists of two parts: a pattern followed by an equivalence string or template. The two parts must be separated by one or more spaces (spaces are not allowed in the parts themselves). The template specifies a username, a host/domain specification,

[4] Usually `PMDF_TABLE:pmdf.cnf`.

[5] Usually `/pmdf/table/pmdf.cnf`.

and the name of a system attached to an existing PMDF channel (the "routing system") to which messages to this address should be sent.

As discussed in Section 2.2.9, additional rewrite rules may be located in an auxiliary database called the domain database. See Section 2.2.9 for information on that database.

The syntax of domain rewriting rules is discussed in further detail below in Section 2.2.4, Patterns and Tags and Section 2.2.5, Templates. First, however, Section 2.2.3 presents a discussion of the application of rewrite rules to addresses, giving an overview of the action of rewrite rules in operation.

2.2.3 Application of Domain Rewriting Rules to Addresses

This section presents a discussion of the operation of domain rewriting rules: how an address is parsed and then transformed via rewrite rules. This section touches briefly on the syntax of rewrite rules as such syntax relates to example addresses, but for full details on rewrite rule syntax, see Section 2.2.4 and Section 2.2.5 below.

There are four steps in the application of the domain rewriting rules to a given address:

1. The first host or domain specification is extracted from the address. (Note that an address may specify more than one host or domain name as is the case with the address jdoe%vax1@example.com.)
2. After extracting the first host or domain name specification, the rewrite rules are scanned for a matching rewrite rule. That is, a search is conducted for a rewrite rule whose pattern portion matches the extracted host/domain name.
3. Once a matching rewrite rule is found, the address is rewritten according to the template portion of that rule. The template also specifies the name of a routing system to which messages to this address should be routed.[7]
4. The routing system name is then compared with the host names associated with each channel. If a match is found, then the message is enqueued to that channel; otherwise, the rewriting process is considered to have failed. If the matching channel is the local channel, then some additional rewriting of the address may occur.

These four steps are described in detail in the following subsections. There are also special template formats which allow for variations in these four steps.

[7] The term "routing system" can be misleading. It does not necessarily mean the name of a system through which the message will be routed but rather is a host name, possibly fictitious, associated with a specific channel.

2.2.3.1 Extraction of the First Host/domain Specification

The process of rewriting an address starts by extracting the first host/domain specification from the address. (Readers who are not familiar with RFC 822 address conventions are advised to read that standard, at least in a cursory fashion, at this point in order to understand the following discussion.) The order in which host/domain specifications in the address are scanned is as follows:

1. Hosts in source routes (read from left to right).
2. Hosts appearing to the right of the at sign.
3. Hosts appearing to the right of the last singleton percent sign.
4. Hosts appearing to the left of the first exclamation point.

The order of the last two items are switched if the `bangoverpercent` keyword is in effect on the channel that is doing the address rewriting. That is, if the channel which is attempting to enqueue the message is itself marked with the `bangoverpercent` channel keyword.[8]

Some highly hypothetical examples of addresses and the host name that would be extracted first are shown below:

Address	First host/domain specification	Comments
user@a	a	a is a "short-form" domain name
user@a.b.c	a.b.c	a.b.c is a "fully-qualified" domain name (FQDN)
user@[0.1.2.3]	[0.1.2.3]	[0.1.2.3] is a "domain literal"
@a:user@b.c.d	a	This is a source-routed address with a a short-form domain name, the "route"
@a.b.c:user@d.e.f	a.b.c	Source routed address, route part is fully-qualified
@[0.1.2.3]:user@d.e.f	[0.1.2.3]	Source-routed address, route part is a domain literal
@a,@b,@c:user@d.e.f	a	Source-routed address with an a to b to c routing
@a,@[0.1.2.3]:user@b	a	Source-routed address with a domain literal in the route part
user%A@B	B	This non-standard form of routing is called a "percent hack"

[8] For instance, if this is a message being sent from a user agent, then the enqueuing channel — the channel doing the rewriting — would be the local, l, channel. If it is a message coming in from Message Router via PMDF-MR then usually it will be the mr_local channel doing the rewriting. And, if it is a message coming in from DECUS UUCP, then the vn_gateway channel will be doing the rewriting.

Address	First host/domain specification	Comments
user%A%B%C@D	D	A built up percent hack
user%A	A	
user%A%B	B	
user%%A%B	B	
user%A%%B	A%%B	Of questionable value
@A:user%B@C	A	
A!user	A	"Bang-style" addressing; commonly used for UUCP
A!user@B	B	
A!user%B@C	C	
A!user%B	B	`nobangoverpercent` keyword active; the default
A!user%B	A	`bangoverpercent` keyword active
@A:B!user@C	A	
@A,@B:C!user%D@E	A	Too grotesque to consider, really

Note that RFC 822 does not say anything about the interpretation of exclamation points, !, and percent signs, %, in addresses. It is customary to interpret percent signs in the same manner as at signs, @, if no at sign is present, so this convention is adopted by PMDF.

The special interpretation of repeated percent signs is used to allow percent signs as part of local usernames, which is used in handling PSIMail and other foreign mail system addresses. The interpretation of exclamation points conforms to RFC 976's "bang-style" address conventions and makes it possible to use UUCP addresses with PMDF.

The order of these interpretations is not specified by either RFC 822 or RFC 976, so the `bangoverpercent` and `nobangoverpercent` keywords can be used to control the order in which they are applied by the channel doing the rewriting. Note that the default is more "standard", although the alternate setting may be useful under some circumstances.

2.2.3.2 Scanning the Rewrite Rules

Once the first host/domain specification has been extracted from the address, PMDF consults the rewrite rules to find out what to do with it. The host/domain specification is compared with the pattern part of each rule (*i.e.,* the left-hand side of each rule). The comparison is case insensitive. Case insensitivity is mandated by RFC 822, UUCP addresses notwithstanding. PMDF is insensitive to case but preserves it whenever possible.

If the host/domain specification does not match any pattern, in which case it is said to "not match any rule", the first part of the host/domain specification — the part before the first period, usually the host name — is removed and replaced with an asterisk and another attempt is made to locate the resulting host/domain specification, but only in

the configuration file rewrite rules (the domain database is not consulted). If this fails the first part is removed and the process is repeated. If this also fails the next part is removed (usually a subdomain) and the rewriter tries again, first with asterisks and then without. All probes that contain asterisks are only done in the configuration file rewrite rules table; the domain database is not checked. This process proceeds until either a match is found or the entire host/domain specification is exhausted. The effect of this procedure is to try to match the most specific domain first, working outward to less specific and more general domains.

A somewhat more algorithmic view of this matching procedure is given below.

0. The host/domain specification is used as the initial value for the comparison strings spec_1 and spec_2. *E.g.,* spec_1 = spec_2 = a.b.c).
1. The comparison string spec_1 is compared with the pattern part of each rewrite rule in the configuration file and then the domain database until a match is found. The matching procedure is exited if a match is found.
2. If no match is found then the leftmost, non-asterisk part of spec_2 is converted to an asterisk. *E.g.,* if spec_2 is a.b.c then it is changed to *.b.c; if spec_2 is *.b.c then it is changed to *.*.c; *etc.* The resulting comparison string spec_2 is compared with *only* the configuration file. The domain database is not consulted. The matching procedure is exited if a match is found.
3. If no match is found then the first part, including any leading period, of the comparison string spec_1 is removed. In the case where spec_1 has only one part (*e.g.,* .c or c), the string is replaced with a single period, ".". If the resulting string spec_1 is of non-zero length, then we return to Step 1. If the resulting string has zero length (*i.e.,* was previously ".") then the lookup process has failed and we exit the matching procedure.

For example, suppose the address iris@sc.cs.example.com is to be rewritten. This causes the rewriter to look for the following patterns in the given order:

Pattern	Files Scanned
sc.cs.example.com	configuration file and then domain database
*.cs.example.com	configuration file rules only
.cs.example.com	configuration file and then domain database
..example.com	configuration file rules only
.example.com	configuration file and then domain database
..*.com	configuration file rules only
.com	configuration file and then domain database
..*.*	configuration file rules only
.	match all rule described in Section 2.2.4.3

Note: Always remember that patterns involving asterisks are only searched for in the configuration file's set of rewrite rules; no searching is done for these patterns in the domain database.

2.2.3.3 Applying the Rewrite Rule Template

Once a host/domain specification matches a rewrite rule, it is rewritten using the template part of the rule. The template specifies three things:

1. a new username for the address,
2. a new host/domain specification for the address, and
3. the name of a system attached to an existing PMDF channel (the "routing system") to which messages to this address should actually be sent.

Template format is discussed in detail in Section 2.2.5. As a quick overview, note that the most common format for templates is A%B@C, where A is the new username, B is the new host/domain specification, and C is the routing system. And the format A@C (which is an abbreviation for A%C@C) is also commonly used.

Substitution strings are allowed in the template. For instance, to mention some of the more commonly used substitution strings, any occurrences of $U in the template are replaced with the username from the original address, any occurrences of $H are replaced with the portion of the host/domain specification that was not matched by the rule, and any occurrences of $D are replaced by the portion of the host/domain specification that was matched by the rewrite rule. Table 2–3 contains a summary of these and other substitution strings which are presented in detail in Section 2.2.6.

As an example, suppose that the host/domain specification adrian@example.com has matched the rewrite rule

```
example.com        $U@EXAMPLE.COM
```

Then the template will produce the username adrian, the host/domain specification EXAMPLE.COM, and the routing system EXAMPLE.COM. In a slightly more complicated example, assume that the host/domain specification has matched the rewrite rule

```
.com               $U%$H$D@TCP-DAEMON
```

In this case, $U = "jdoe", $H = "example", and $D = ".com". The template produces the username adrian, the host/domain specification example.com, and the routing system TCP-DAEMON.

2.2.3.4 Finishing the Rewriting Process

One of two things can happen once the host/domain specification is rewritten. If the routing system is not associated with the local channel, or associated a channel marked with the `routelocal` channel keyword, or there are no additional host/domain specifications in the address, then the rewritten specification is substituted into the address replacing the original specification that was extracted for rewriting, and the rewriting process terminates.

If the routing system matches the local channel (or a channel marked with the `routelocal` channel keyword) and there are additional host/domain specifications that appear in the address, then the rewritten address is discarded, the original (initial) host/domain specification is removed from the address, a new host/domain specification is extracted from the address and the entire process is repeated. Rewriting will continue

until either all the host/domain specifications are gone or a route through a non-local, non-`routelocal` channel is found. This iterative mechanism is PMDF's way of providing support for source routing. In effect, superfluous routes through the "local system" are removed from addresses by this process.

2.2.3.5 Rewrite Rule Failure

If a host/domain specification fails to match any rewrite rule and no default rule is present, PMDF simply uses the specification "as-is"; *i.e.,* the original specification becomes both the new specification and the routing system. If the address has a nonsensical host/domain specification it will be detected when the routing system does not match any system name associated with any channel. This relaxed interpretation of rewrite rule failures allows isolated PMDF sites that only communicate with a small number of systems to get by without any rewrite rules whatsoever.

2.2.3.6 Syntax Checks After Rewriting

No additional syntax checking is done after the rewrite rules have been applied to an address. This laxity is deliberate — it makes it possible for rewrite rules to be used to convert addresses into formats that do not conform to RFC 822. However, this also means that mistakes in the configuration file may result in messages leaving the PMDF system with incorrect or illegal addresses.

2.2.3.7 Handling of Domain Literals

Domain literals are handled specially during the rewriting process. If a domain literal appearing in the domain portion of an address does not match a rewrite rule pattern as-is, the literal is interpreted as a group of strings separated by periods and surrounded by square brackets.[9] The rightmost string is removed and the search is repeated. If this does not work the next string is removed, and so on until only empty brackets are left. If the search for empty brackets fails, the entire domain literal is removed and rewriting proceeds with the next section of the domain address, if there is one. No asterisks are used in the internal processing of domain literals; when an entire domain literal is replaced by an asterisk the number of asterisks corresponds to the number of elements in the domain literal.

Like normal domain/host specifications, domain literals are also tried in most specific to least specific order. The first rule whose pattern matches will be the one used to rewrite the host/domain specification. If there are two identical patterns in the rules list, the one which appears first will be used.

As an example, suppose the address iris@[198.162.3.40] is to be rewritten. The rewriter looks for [198.162.3.40], then [198.162.3.], then [198.162.], then [198.], then [ ], then [*.*.*.*], and finally the match-all rule ".".

[9] Note that the support of numeric domain literals is not required by either PMDF or RFC 822.

When domain literals are combined with domain names the number of lookup attempts gets to be quite large. This is *not* normal usage and its use is *strongly discouraged*. For example, the address iris@[1.2].a.[3.4].b would generate requests for:

```
[1.2].a.[3.4].b
[1.].a.[3.4].b
[].a.[3.4].b
[*.*].a.[3.4].b
.a.[3.4].b
[*.*].*.[3.4].b
.[3.4].b
[*.*].*.[3.].b
.[3.].b
[*.*].*.[].b
.[].b
[*.*].*.[*.*].b
.b
[*.*].*.[*.*].*
.
```

2.2.4 Patterns and Tags

Most rewrite rule patterns consist either of a specific host name that will match only and exactly that host, *e.g.*,

```
host.domain.com
```

or consist of a subdomain pattern that will match any host/domain in the entire subdomain, *e.g.*,

```
.domain.com
```

A rewrite rule pattern such as the above would match any host.domain.com or host.subnet.domain.com sort of host/domain name. Note, however, that it will *not* match the exact host name domain.com; to match the exact host name domain.com, a separate domain.com pattern would be needed.

Since as discussed in Section 2.2.3.2 PMDF attempts to rewrite host/domain names starting from the specific host name and then incrementally generalizing the name to make it less specific, this means that a more specific rewrite rule pattern will be preferentially used over more general rewrite rule patterns. For instance, if the rewrite rule patterns

```
hosta.subnet.domain.com
.subnet.domain.com
.domain.com
```

are present in the configuration file,[a] then an address of jdoe@hosta.subnet.domain.com will match the specific hosta.subnet.domain.com rewrite rule pattern, while an address of jdoe@hostb.subnet.domain.com will match the more general .subnet.domain.com rewrite

[a] Note that the order of these rewrite rule patterns does not strictly matter, though a more specific to less specific ordering is commonly used for efficiency and esthetic reasons.

rule pattern, and an address of jdoe@hostc.domain.com will match the .domain.com rewrite rule pattern.

In particular, the use of rewrite rules incorporating subdomain rewrite rule patterns is common for sites on the Internet. Such a site will typically have a number of rewrite rules for their own internal hosts and subnets, and then will include rewrite rules for the top-level Internet subdomains into their configuration from the file `internet.rules` stored in the PMDF table directory. The presence of such

```
!    Ascension Island
.AC                                          $U%$H$D@TCP-DAEMON
. [text
.    removed for
.              brevity]
!    Zimbabwe
.ZW                                          $U%$H$D@TCP-DAEMON
```

rewrite rules, with rewrite rule patterns that match the top level Internet domains and rewrite rule templates that rewrite addresses matching such patterns to an outgoing TCP/IP channel, ensure that messages to Internet destinations (other than to the "internal" host destinations handled via more specific rewrite rules) will be properly rewritten and routed out an outgoing TCP/IP channel.

IP domain literals follow a similar hierarchical matching pattern, though with right-to-left (rather than left-to-right) matching. For instance, the pattern

```
[1.2.3.4]
```

matches only and exactly the IP literal [1.2.3.4], while

```
[1.2.3.]
```

matches anything in the 1.2.3.0 subnet.

In addition to the more common sorts of host or subdomain rewrite rule patterns discussed above, rewrite rules may also make use of several special patterns, summarized in Table 2–1, and discussed in the following subsections.

Table 2–1 Summary of Special Patterns for Rewrite Rules

Pattern	Name	Section	Usage
$%	Percent hack rule	2.2.4.1	Matches any host/domain specification of the form A%B.
$!	Bang-style rule	2.2.4.2	Matches any host/domain specification of the form B!A.
[]	IP literal match-all rule	2.2.3.7	Match any IP domain literal.
.	Match-all rule	2.2.4.3	Matches any host/domain specification.

In addition to these special patterns, PMDF also has the concept of "tags" which may appear in rewrite rule patterns. These tags are used in situations where an address may be rewritten several times and, based upon previous rewritings, distinctions must be made in subsequent rewritings by controlling which rewrite rules match the address.

2.2.4.1 A Rule to Match Percent Hacks

If PMDF tries to rewrite an address of the form A%B and fails, it tries one extra rule before falling through and treating this address form as A%B@localhost. This extra rule is the percent hack rule. The pattern is $%. The pattern never changes. This rule is only activated when a local part containing a percent sign has failed to rewrite any other way (including the match all rule described below).

The percent hack rule is useful for assigning some special, internal meaning to percent hack addresses.

2.2.4.2 A Rule to Match Bang-style (UUCP) Addresses

If PMDF tries to rewrite an address of the form B!A and fails, it tries one extra rule before falling through and treating this address form as B!A@localhost. This extra rule is the bang-style rule. The pattern is $!. The pattern never changes. This rule is only activated when a local part containing an exclamation point has failed to rewrite any other way (including the default rule described below).

The bang-style rule can be used to force UUCP style addresses to be routed to a system with comprehensive knowledge of UUCP systems and routing.

2.2.4.3 A Rule to Match Any Address

The special pattern "." (a single period) will match *any* host/domain specification if no other rule matches and the host/domain specification cannot be found anywhere in the channel table. In other words, the "." rule is used as a last resort when address rewriting would fail otherwise.

Using this very general rule can simplify some PMDF installations at the expense of propagating possibly bogus addresses. This special default rule should only be used when PMDF does not have complete routing information available and has to defer judgment of address validity to another system or systems. Care should be taken to insure that such unchecked addresses are only sent to systems that are capable of handling them.

Note: When the match-all rule matches and its template is expanded, $H expands to the full host name and $D expands to a single dot ".". Thus, $D is of limited use in a match-all rule template!

2.2.4.4 Tagged Rewrite Rule Sets

As the rewrite process proceeds it may be appropriate to bring different sets of rules into play. This is accomplished by the use of the rewrite rule tag. The current tag is prepended to each pattern before looking it up in the configuration file or domain database. The tag can be changed by any rewrite rule that matches by using the $T substitution string in the rewrite rule template (described below).

Tags are somewhat sticky; once set they will continue to apply to all hosts that are extracted from a single address. This means that care must be taken to provide alternate rules that begin with the proper tag values once any tags are used. In practice this is rarely a problem since tags are usually used in only very specialized applications. Once the rewriting of the address is finished the tag is reset to the default tag — an empty string.

By convention all tag values end in a vertical bar | . This character is not used in normal addresses and thus is free to delineate tags from the rest of the pattern.

See Section 2.2.6.14 for an example of using tagged rewrite rules.

2.2.5 Templates

Once a host/domain specification matches a rewrite rule, it is rewritten using the template part of the rule. The template specifies three things:

1. a new username for the address,
2. a new host/domain specification for the address, and
3. the name of a system attached to an existing PMDF channel (the “routing system”) to which messages to this address should actually be sent.

A summary of the template formats for rewrite rules is presented in Table 2–2. The substitution strings and control sequences which may be used with templates are discussed in Section 2.2.6.

Table 2–2 Summary of Template Formats for Rewrite Rules

Template	Section	Usage
A%B	2.2.5.2	A becomes the new user/mailbox name, B becomes the new host/domain specification, rewrite again
A@B	2.2.5.1	Treated as A%B@B
A%B@C	2.2.5.1	A becomes the new user/mailbox name, B becomes the new host/domain specification, route to C
A@B@C	2.2.5.3	Treated as A@B@C@C.
A@B@C@D	2.2.5.3	A becomes the new user/mailbox name, B becomes the new host/domain specification, insert C as a source route, route to D

Other formats, such as A%B%C and so forth, are reserved for the implementation of future capabilities in PMDF and should not be used as their function may change in a future release.

2.2.5.1 Ordinary Rewriting Templates, A@B or A%B@C

The most commonly used form of rewrite rule template is A%B@C, where A is the new username, B is the new host/domain specification, and C is the routing system (official channel name). If B and C are identical, %B may be omitted; *i.e.*, you may simply use A@C when B and C are identical.

2.2.5.2 Repeated Rewritings Template, A%B

The special rewrite rule template format A%B is used for "meta-rules" that require additional rewriting after their application. When an A%B pattern is encountered, A becomes the new username and B becomes the new host/domain specification, and then the entire rewriting process is repeated on the resulting new address. All other rewrite rule formats cause the rewriting process to terminate after the rule has been applied.

For example, the rule

```
.removeable       $U%$H
```

has the effect of removing all occurrences of the `.removeable` domain from the ends of addresses.

Extreme care must be taken when using these repeating rules; careless use can create a "rules loop" that will hang PMDF in an infinite loop. For this reason meta-rules should only be used when absolutely necessary. Be sure to test them with the OpenVMS command PMDF TEST/REWRITE or the UNIX or NT command `pmdf test -rewrite`.

2.2.5.3 Specified Route Rewriting Templates, A@B@C or A@B@C@D

The special rewrite rule template format A@B@C works in the same way as the usual A%B@C rule, except that the routing system C will also be inserted into the address as a source route. This inclusion of the routing system in the address may be needed by some channels that have to establish a connection to the routing system and determine the name of the routing system from the envelope To: address. For instance, the rewrite rule

```
vax1      $U@vax1@example.com
```

would rewrite the address jdoe@vax1 into the source routed address @example.com:jdoe@vax1. The routing system will be example.com.

The template format A@B@C@D uses A as the new username, B is the new host/domain specification, C is inserted as a source route, and D is the routing system. This is the most general template format available.

Note: Channel table rewriting may change the name of the routing system if the address being rewritten is an envelope To: address. See Section 2.3.3 for further information on channel table rewriting.

2.2.5.4 Case Sensitivity in Rewrite Rule Templates

Unlike the patterns in rewrite rules, character case in templates is preserved. This is necessary when using rewrite rules to provide an interface to a mail system such as UUCP which is sensitive to character case. Note that substitution sequences like $U and $D that substitute material extracted from addresses also preserve the original case of characters.

When it is desirable to force substituted material to use a particular case, for instance, to force mailboxes to lowercase on UNIX systems, special substitution sequences can be used in templates to force substituted material to a desired case. Specifically, $\ forces subsequent substituted material into lower case, $^ forces subsequent substituted material into upper case, and $_ says to use the original case. So you can use a rule such as

```
unix.example.com     $\$U$_%unix.example.com
```

to force mailboxes to lowercase for unix.example.com addresses.

2.2.6 Template Substitutions and Rewrite Rule Control Sequences

Substitutions are used to substitute into the rewritten address a character string the value of which is determined by the particular substitution sequence used. For instance in the template

```
$U@example.com
```

the `$U` is a substitution sequence. It causes the username portion of the address being rewritten to be substituted into the output of the template. Thus, if jdoe@vax1.example.com was being rewritten by this template, the resulting output would be jdoe@example.com, the $U substituting in the username portion, jdoe, of the original address.

Special control sequences may also appear in rewrite rule templates. These sequences impose additional conditions to the applicability of a given rewrite rule: not only must the pattern portion of the rewrite rule match the host/domain specification being examined, but other aspects of the address being rewritten must meet conditions set by the control sequence or sequences. For instance, the $E control sequence requires that the address being rewritten be an envelope address while the $F sequence requires that it be a forward pointing address. Thus, the rewrite rule

```
example.com     $U@example.com$E$F
```

will only apply to (*i.e.,* only rewrite) envelope To: addresses of the form user@example.com. If a domain/host specification matches the pattern portion of a rewrite rule but doesn't meet all of the criteria imposed by control sequences in the rule's template, then the rewrite rule fails and the rewriter continues to look for other applicable rules. This makes possible sets of rewrite rules such as

```
example.com     $U%example.com@directory-daemon
example.com     $U@example.com$Ndirectory
```

which will result in messages to user@example.com being passed to the directory channel. However, should the directory channel rewrite a message with the address user@example.com, that message will not again pass through the directory channel. This then allows all mail to user@example.com to pass through the directory channel and for the directory channel to emit mail to that address without causing a mail loop.

A summary of template substitutions and control sequences is presented in Table 2–3.

Table 2–3 Summary of Template Substitutions and Control Sequences

Substitution sequence	Section	Substitutes
$D	2.2.6.2	Portion of domain specification that matched
$H	2.2.6.2	Unmatched portion of host/domain specification; left of dot in pattern
$L	2.2.6.2	Unmatched portion of domain literal; right of dot in pattern literal
$U	2.2.6.1	Username from original address
$0U	2.2.6.1	Local part (username) from original address, minus any subaddress
$1U	2.2.6.1	Subaddress, if any, from local part (username) of original address
$$	2.2.6.3	Inserts a dollar sign (literal)
$%	2.2.6.3	Inserts a percent sign (literal)
$@	2.2.6.3	Inserts an at sign (literal)
$\	2.2.5.4	Force substituted material to lowercase
$^	2.2.5.4	Force substituted material to uppercase
$_	2.2.5.4	Use original case
$W	2.2.6.9	Substitutes in a random, unique string
$]...[	2.2.6.4	LDAP search URL lookup
$(*text*)	2.2.6.5	General database substitution; rule fails if lookup fails
${...}	2.2.6.6	Apply specified mapping to supplied string
$[...]	2.2.6.7	Invoke customer supplied routine; substitute in result
$&n	2.2.6.8	nth part of unmatched (or wildcarded) host as counting from left to right starting from 0
$!*n*	2.2.6.8	*n*th part of unmatched (wildcarded) host as counted from right to left starting from 0
$**n*	2.2.6.8	*n*th part of matching pattern as counting from left to right starting from 0
$#*n*	2.2.6.8	*n*th part of matching pattern as counted from right to left starting from 0
$*n*D	2.2.6.2	Portion of domain specification that matched, preserving from the *n*th leftmost part starting from 0
$*n*H	2.2.6.2	Portion of host/domain specification that didn't match, preserving from the *n*th leftmost part starting from 0

Control sequence	Section	Effect on rewrite rule
$E	2.2.6.12	Apply only to envelope addresses
$B	2.2.6.12	Apply only to header/body addresses
$F	2.2.6.12	Apply only to forward-directed (*e.g.,* To:) addresses
$R	2.2.6.12	Apply only to backwards-directed (*e.g.,* From:) addresses

Table 2–3 (Cont.) Summary of Template Substitutions and Control Sequences

Control sequence	Section	Effect on rewrite rule
$M*channel*	2.2.6.10	Apply only if channel *channel* is rewriting the address
$N*channel*	2.2.6.10	Fail if channel *channel* is rewriting the address
$Q*channel*	2.2.6.11	Apply if sending to channel *channel*
$C*channel*	2.2.6.11	Fail if sending to channel *channel*
$S	2.2.6.13	Apply if host is from a source route
$A	2.2.6.13	Apply if host is to the right of the at sign
$P	2.2.6.13	Apply if host is to the right of a percent sign
$X	2.2.6.13	Apply if host is to the left of an exclamation point
$T*newtag*	2.2.6.14	Set the rewrite rule tag to *newtag*
$*n*?*errmsg*	2.2.6.15	If rewriting fails return *errmsg* instead of the default error message

2.2.6.1 Username and Subaddress Substitution, $U, $0U, $1U

Any occurrences of $U in the template are replaced with the username (local part) from the original address. Note that usernames of the form a."b" will be replaced by "a.b" as current Internet standardization work is deprecating the former syntax from RFC 822 and it is expected that the latter usage will become mandatory in future.

Any occurrences of $0U in the template are replaced with the username from the original address, minus any subaddress (and subaddress indication character such as +). Any occurrences of $1U in the template are replaced with the subaddress and subaddress indication character, if any, from the original address. (See Section 2.3.4.71 and Section 3.1.1.6 for background on subaddresses.) So note that $0U and $1U are complementary pieces of the username, with $0U$1U being equivalent to a simple $U.

$0U and $1U are most commonly used in PMDF MessageStore rewrite rules, where it is common to force the account portion of the local part to lowercase while retaining original case in the subaddress since the subaddress in a PMDF MessageStore address indicates a folder name. For instance, a rewrite rule:

```
msgstore.example.com    $\$0U$_$1U@msgstore.example.com
```

will cause an address such as nAmE@msgstore.example.com to be transformed (rewritten) to name@msgstore.example.com, while an address such as nAmE+sUbAdDrEsS@msgstore.example.com would be transformed to name+sUbAdDrEsS@msgstore.example.com.

2.2.6.2 Host/domain and IP Literal Substitutions, $D, $H, $*n*D, $*n*H, $L

Any occurrences of $H are replaced with the portion of the host/domain specification that was not matched by the rule. Any occurrences of $D are replaced by the portion of the host/domain specification that was matched by the rewrite rule. $*n*H and $*n*D are variants that preserve the normal $H or $D portion from the *n*th leftmost part starting counting from 0. Or another way of putting it is that $*n*H and $*n*D omit the leftmost *n* parts (starting counting from 1) of what would normally be a $H or $D, substitution, respectively. In particular, $0H is equivalent to $H and $0D is equivalent to $D.

For example, suppose the address jdoe@host.example.com matches the rewrite rule

```
host.example.com     $U%$1D@TCP-DAEMON
```

Then the result of the rewrite rule will be jdoe@example.com with TCP-DAEMON used as the outgoing channel. Here where $D would have substituted in the entire domain that matched, host.example.com, the $1D instead substitutes in the portions of the match starting from part 1 (part 1 being "example"), so substitutes in example.com.

$L substitutes the portion of a domain literal that was not matched by the rewrite rule.

2.2.6.3 Literal Character Substitutions, $$, $%, $@

The $, %, and @ characters are normally metacharacters in rewrite rule templates. To insert a literal such character, quote it with a dollar character, $. *I.e.*, $$ expands to a single dollar sign, $; $% expands to a single percent, % (the percent is not interpreted as a template field separator in this case); and $@ expands to a single at sign, @ (also not interpreted as a field separator).

2.2.6.4 LDAP Query URL Substitutions, $]...[

A substitution of the form `$]`*`ldap-url`*`[` is handled specially. *`ldap-url`* is interpreted as an LDAP query URL and the result of the LDAP query is substituted. Standard LDAP URLs are used, with the host and port omitted; the host and port are instead specified with the LDAP_HOST and LDAP_PORT PMDF options (see Section 7.3.2 for further discussion of this option). That is, the LDAP URL should be specified as

```
ldap:///dn[?attributes[?scope?filter]]
```

where the square bracket characters *`[`* and *`]`* shown above indicate optional portions of the URL. The *`dn`* is required and is a distinguished name specifying the search base. The optional *`attributes`*, *`scope`*, and *`filter`* portions of the URL further refine what information to return. For a rewrite rule, the desired *`attributes`* to specify returning might be a `mailRoutingSystem` attribute (or some similar attribute). The *`scope`* may be any of `base` (the default), `one`, or `sub`. And the desired *`filter`* might be to request the return of the object whose `mailDomain` value matches the domain being rewritten.

For instance, at a site example.com with an LDAP server running on port 389 of the system ldap.example.com, the PMDF option file might have the lines

```
LDAP_HOST=ldap.example.com
LDAP_PORT=389
```

set, and if the LDAP directory schema includes attributes `mailRoutingSystem` and `mailDomain`, then a possible rewrite rule to determine to which system to route a given sort of address might appear as:

```
.example.com  \
    $U%$H$D@$]<ldap:///o=example.com?mailRoutingSystem?sub?(mailDomain=$D)[
```

where here the LDAP URL substitution sequence `$D` is used to substituted in the current domain name into the LDAP query constructed; for ease in reading, the backslash character, `\`, is used to continue the single logical rewrite rule line onto a second physical line. See Table 3–1 for a full list of the LDAP URL substitution sequences available.

2.2.6.5 General Database Substitutions, $(...)

A substitution of the form `$(`*text*`)` is handled specially. The *text* part is used as a key to access the special database PMDF_GENERAL_DATABASE.[b] This database is generated with the `pmdf crdb` utility. If *text* is found in the database the corresponding template from the database is substituted. If *text* does not match an entry in the database the rewrite process fails; it is as if the rewrite rule never matched in the first place. If the substitution is successful the template extracted from the database is re-scanned for additional substitutions. However, additional `$(`*text*`)` substitutions from the extracted template are prohibited in order to prevent endless recursive references.

As an example, suppose that the address jdoe@example.decnet matches the rewrite rule

```
.DECNET      $($H)
```

Then, the text string "example" will be looked up in the general database and the result of the look up, if any, instead used for the rewrite rule's template. Suppose that the result of looking up "example" is "$u%examplevax.example.com@decnet". Then the output of the template will be jdoe@examplevax.example.com (*i.e.,* username = jdoe, host/domain specification = examplevax.example.com), and the routing system will be decnet.

If a general database exists it should be world readable to insure that it operates properly.

2.2.6.6 Apply Specified Mapping, ${...}

A substitution of the form `${`*mapping*`,`*argument*`}` is handled specially. The *mapping*`,`*argument* part is used to find and apply a mapping from the PMDF mapping file. The *mapping* field specifies the name of the mapping table to use while *argument* specifies the string to pass to the mapping. The mapping must exist and must set the $Y flag in its output if it is successful; if it doesn't exist or doesn't set $Y the rewrite will fail. If successful the result of the mapping is merged into the template at the current location and reexpanded.

This mechanism allows PMDF's rewriting process to be extended in various complex ways. For example, the username part of an address can be selectively analyzed and modified, which normally isn't a feature PMDF's rewriting process is capable of.

[b] On OpenVMS systems, this database is the file pointed at by the logical name PMDF_GENERAL_DATABASE, which is typically the file `PMDF_TABLE:general.dat`; on UNIX systems, this database consists of the file specified by the PMDF_GENERAL_DATABASE option in the `/etc/pmdf_tailor` file, which is usually the file `/pmdf/table/generaldb.*`.

2.2.6.7 Customer-supplied Routine Substitutions, $[...]

A substitution of the form `$[`*`image`*`,`*`routine`*`,`*`argument`*`]` is handled specially. The *`image`*`,`*`routine`*`,`*`argument`* part is used to find and call a customer-supplied routine. At run-time on OpenVMS, PMDF uses `LIB$FIND_IMAGE_SYMBOL` to dynamically load and link to the routine *`routine`* from the shareable image *`image`*; at run-time on UNIX, PMDF uses `dlopen` and `dlsym` to dynamically load and call the routine *`routine`* from the shared library *`image`*. The routine *`routine`* is then called as a function with the following argument list:

status = *`routine`* (**argument**, **arglength**, **result**, **reslength**)

argument and **result** are 252 byte long character string buffers. On OpenVMS **argument** and **result** are passed by descriptor (a class S descriptor is used to insure maximum compatibility); on UNIX and Windows, **argument** and **result** are passed as a pointer to a character string, (*e.g.*, in C, as char*). **arglength** and **reslength** are signed, long integers passed by reference. On input, **argument** contains the *`argument`* string from the rewrite rule template, and **arglength** the length of that string. On return, the resultant string should be placed in **result** and its length in **reslength**. This resultant string will then replace the "`$[`*`image`*`,`*`routine`*`,`*`argument`*`]`" in the rewrite rule template. The routine *`routine`* should return 0 if the rewrite rule should fail and -1 if the rewrite rule should succeed.

This mechanism allows PMDF's rewriting process to be extended in all sorts of complex ways. For example, a call to some type of name service could be performed and the result used to alter the address in some fashion. For instance, directory service lookups for forward pointing addresses (*e.g.,* To: addresses) to the host example.com might be performed as follows with the following rewrite rule (the $F, described in Section 2.2.6.12 causes this rule to only be used for forward pointing addresses):

```
example.com       $F$[LOOKUP_IMAGE,LOOKUP,$U]
```

A forward pointing address jdoe@example.com will, when it matches this rewrite rule, cause `LOOKUP_IMAGE` (which is a shareable image on OpenVMS and a shared library on UNIX) to be loaded into memory, and then cause the routine `LOOKUP` called with "jdoe" as the **argument** parameter. The routine `LOOKUP` might then return a different address, say, John.Doe%vax.example.com in the **result** parameter and the value -1 to indicate that the rewrite rule succeeded. The percent sign in the result string causes, as descibed in Section 2.2.5.2 the rewriting process to start over again using John.Doe@vax.example.com as the address to be rewritten.

VMS

On OpenVMS systems, since `LIB$FIND_IMAGE_SYMBOL` is used to dynamically load the site-supplied image *`image`*, then *`image`* must be a logical name pointing to the actual shareable image. Moreover, as this mechanism will be invoked by PMDF in a variety of contexts, the logical must be an executive mode logical, any logicals it references must also be executive mode logicals, and the image itself must be world readable and installed as a known image.

UNIX

On UNIX systems, the site-supplied shared library image *`image`* should be world readable.

Note: This facility is not designed for use by casual users; it is intended to be used to extend PMDF's capabilities system-wide.

2.2.6.8 Single Field Substitutions, $&, $!, $*, $#

Single field substitutions extract a single subdomain part from the host/domain specification being rewritten. The available single field substitutions are shown in Table 2–4.

Table 2–4 Single Field Substitutions

Control Sequence	Usage
$&n	Substitute the nth element, n=0,1,2,..,9, in the host specification (the part that did not match or matched a wildcard of some kind). Elements are separated by dots; the first element on the *left* is element zero. The rewrite fails if the requested element does not exist.
$!n	Substitute the nth element, n=0,1,2,..,9, in the host specification (the part that did not match or matched a wildcard of some kind). Elements are separated by dots; the first element on the *right* is element zero. The rewrite fails if the requested element does not exist.
$*n	Substitute the nth element, n=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the *left* is element zero. The rewrite fails if the requested element does not exist.
$#n	Substitute the nth element, n=0,1,2,...,9, in the domain specification (the part that did match explicit text in the pattern). Elements are separated by dots; the first element on the *right* is element zero. The rewrite fails if the requested element does not exist.

Suppose the address jdoe@vaxa.example.com matches the rewrite rule

```
*.EXAMPLE.COM      $U%$&0.example.com@mailhub.example.com
```

Then the result from the template will be jdoe@vaxa.example.com with mailhub.example.com used as the routing system.

2.2.6.9 Unique String Substitutions

Each use of the $W control sequence inserts a text string composed of upper case letters and numbers that is designed to be unique and unrepeatable. $W is useful in situations where nonrepeating address information must be constructed.

2.2.6.10 Source Channel-specific Rewrite Rules, $M, $N

It is possible to have rewrite rules that act only in conjunction with specific source channels. This is useful when a shortform name has two meanings, one when it appears in a message arriving on one channel and another when it appears in a message arriving on a different channel.

Source channel-specific rewriting is associated with the channel program in use and the channel keywords `rules` and `norules`. If `norules` is specified on the channel associated with a PMDF component that is doing the rewriting, no channel-specific rewrite checking is done. If `rules` is specified on the channel, channel-specific rule checks are enforced. `rules` is the default.

Source channel-specific rewriting is *not* associated with the channel a given address matches. It depends only on the PMDF component doing the rewriting and that component's channel table entry.

Channel-specific rewrite checking is triggered by the presence of a $N or $M control sequence in the template part of a rule. The characters following the $N or $M, up until either an at sign, percent sign, or subsequent $N, $M, $Q, $C, $T, or $? are interpreted as a channel name.

$M*channel* causes the rule to fail if the channel *channel* is not currently doing the rewriting. $N*channel* causes the rule to fail if the channel *channel* is doing the rewriting.

Multiple $M and $N clauses may be specified. If any one of multiple $M clauses matches, the rule will succeed. If any of multiple $N clauses matches, the rule will fail.

For example, suppose that the shortform host name ACUVAX is both a local DECnet host and a BITNET host. For local use, it probably makes sense for any use of ACUVAX to map to the DECnet host. But for messages coming in on the BITNET channel, interpreting this name as the BITNET host would be more appropriate.

This problem might be solved with rewrite rules of the form given below; the use of Jnet and not ANJE is assumed in this example.

```
acuvax            $U%acuvax.bitnet@Jnet-DAEMON$Mbit_local
acuvax.bitnet     $U%acuvax.bitnet@Jnet-DAEMON
acuvax            $U%acuvax.decnet@decnet-mail
acuvax.decnet     $U%acuvax.decnet@decnet-mail
```

These rewrite rules produce the following behavior: traffic for the host ACUVAX on the bit_local channel and traffic for the host ACUVAX.BITNET are handled by the bit_local channel; other traffic for the host ACUVAX and the host ACUVAX.DECNET is handled by the decnet-mail channel.

2.2.6.11 Destination Channel-specific Rewrite Rules, $C, $Q

It is possible to have rewrite rules whose application is dependent upon the channel to which a message is being enqueued. This is useful when there are two names for some host, one known to one group of hosts and one known to another. By using different channels to send mail to each group, addresses can be rewritten to refer to the host under the name known to each group.

Destination channel-specific rewriting is associated with the channel to which a message is being enqueued and the channel keywords `rules` and `norules` on that channel. If `norules` is specified on the destination channel, no channel-specific rewrite checking is done. If `rules` is specified on the destination channel, channel-specific rule checks are enforced. `rules` is the default.

Destination channel-specific rewriting is *not* associated with the channel a given address matches. It depends only on the message's envelope To: address. When a message is enqueued, its envelope To: address is first rewritten to determine to which channel the message will be enqueued. *During the rewriting of the envelope To: address any $C and $Q control sequences are ignored.* Once the envelope To: address is rewritten and the destination channel determined, then the $C and $Q control sequences are honored as other addresses associated with the message are rewritten.

Destination channel-specific rewrite checking is triggered by the presence of a $C or $Q control sequence in the template part of a rule. The characters following the $C or $Q, up until either an at sign, percent sign, or subsequent $N, $M, $C, $Q, $T, or $? are interpreted as a channel name.

$Q*channel* causes the rule to fail if the channel *channel* is not the destination. $C*channel* causes the rule to fail if the channel *channel* is the destination.

Multiple $Q and $C clauses may be specified. If any one of multiple $Q clauses matches, the rule will succeed. If any of multiple $C clauses matches, the rule will fail.

For example, suppose the local host's TCP/IP channel used to communicate with the Internet is the ptcp_local channel. Then, to prevent "raw" user@host.bitnet style addresses from appearing on messages queued to that channel, a rewrite rule of the form

```
.BITNET        $U$%$H$D@interbit.cren.net$Qptcp_local
```

might be used. This will, in messages destined to the ptcp_local channel, transform addresses of the form user@host.bitnet to user%host.bitnet@interbit.cren.net.

2.2.6.12 Direction and Location-specific Rewrites, $B, $E, $F, $R

It is sometimes useful to specify rewrite rules that only apply to envelope addresses or, alternately, only apply to header addresses. The control sequence $E forces a rewrite to fail if the address being rewritten is not an envelope address. The control sequence $B forces a rewrite to fail if the address being rewritten is not from the message header or body. These sequences have no other effects on the rewrite and may appear anywhere in the rewrite rule template.

Addresses may also be categorized by direction. A forward-pointing address is one that originates on a To:, Cc:, Resent-to:, or other header or envelope line that refers to a destination. A backwards-pointing address is something like a From:, Sender:, or Resent-From:, which refers to a source. The control sequence $F causes the rewrite to fail if the address is backwards-pointing. The control sequence $R causes the rewrite to fail if the address is forward-pointing.

The following rewrite rule causes forward pointing envelope addresses (*i.e.,* envelope To: addresses) of the form user@host.decnet to be rewritten to user@host and the message routed to the channel associated with the host decnet-mail:

```
.decnet                 $U%$H@decnet-mail$E$F
.decnet                 $U@$H.example.com
```

All other addresses of the form user@host.decnet are rewritten to user@host.example.com by the second rewrite rule.

2.2.6.13 Host Location-specific Rewrites, $A, $P, $S, $X

Circumstances occasionally require rewriting that's sensitive to the location where a host name appears in an address. Host names can appear in several different contexts in an address: in a source route, to the right of the at sign, to the right of a percent sign in the local-part, or to the left of an exclamation point in the local-part. Under normal circumstances a host name should be handled in the same way regardless of where it appears. Situations can arise, however, which may necessitate specialized handling.

Four control sequences are used to control matching on the basis of the host's location in the address. $S specifies that the rule may match a host extracted from a source route, $A specifies that the rule may match a host found to the right of the at sign, $P specifies that the rule may match a host found to the right of a percent sign, and $X specifies that the rule may match a host found to the left of an exclamation point. The rule will fail if the host is from a location other than one specified.

These sequences can be combined in a single rewrite rule. For example, if $S and $A are specified the rule will match hosts specified in either a source route or to the right of the at sign. Specifying none of these sequences is equivalent to specifying all of them; the rule can match regardless of location.

2.2.6.14 Changing the Current Tag Value, $T

The $T control sequence is used to change the current rewrite rule tag. The rewrite rule tag is prepended to all rewrite rule patterns before they are looked up in the configuration file and domain database. Text following the $T, up until either an at sign, percent sign, $N, $M, $Q, $C, $T, or $? is taken to be the new tag.

Tags are useful in handling special addressing forms where the entire nature of an address is changed when a certain component is encountered. For example, suppose that the special host name `internet`, when found in a source route, should be removed from the address and the resulting address forcibly matched against the TCP-DAEMON channel. This could be implemented with rules like the following (`localhost` is assumed to be the official name of the local host):

```
internet                $S$U@localhost$Tmtcp-force|
mtcp-force|.            $U%$H@TCP-DAEMON
```

The first rule will match the special host name `internet` if it appears in the source route. It forcibly matches `internet` against the local channel, which insures that it will be removed from the address. A rewrite tag is then set. Rewriting proceeds, but no regular rule will match because of the tag. Finally, the default rule is tried with the tag, and the second rule of this set fires, forcibly matching the address against the TCP-DAEMON channel regardless of any other criteria.

2.2.6.15 Controlling Error Messages Associated with Rewriting, $?

PMDF provides default error messages when rewriting and channel matching fail. The ability to change these messages can be useful under certain circumstances. For example, if someone tries to send mail to an ethernet router box, it may be considered more informative to say something like "our routers cannot accept mail" rather than the usual "illegal host/domain specified". A special control sequence can be used to change the error message that will be printed if the rule fails. The sequence $? is used to specify an error message. Text following the $?, up until either an at sign, percent sign, $N, $M, $Q, $C, $T, or $? is taken to be the text of the error message to print if the result of this rewrite fails to match any channel. The setting of an error message is "sticky" and will last through the rewriting process.

A rule that contains a $? operates just like any other rule. The special case of a rule containing only a $? and nothing else receives special attention — the rewriting process is terminated without changing the mailbox or host portions of the address and the host is looked up as-is in the channel table. This lookup is expected to fail and the error message will be returned as a result.

For instance, if the final rewrite rule in the PMDF configuration file is

```
.       $?Unrecognized address; contact postmaster@xyzzy.com
```

then any unrecognized host/domain specifications which will fail will, in the process of failing, generate the error message "Unrecognized address; contact postmaster@xyzzy.com".

There is an optional value that may be specified between the $ and the ?. This value tells PMDF what error number to use with the specified error message. The default if no value is specified is 5.1.2, a permanent error. You can use the optional value to specify a temporary error (4.y.z) or a different permanent error.

The value is formatted as follows: to get the error number x.y.z, the value between the $ and ? should be x00y00z. For example:

```
.       $4001002?Unrecognized address; contact postmaster@xyzzy.com
```

Generates a 400-level error response with the error message "4.1.2 Unrecognized address; contact postmaster@xyzzy.com".

2.2.7 Rewrite Rules Example

The following example provides some sample rewrite rules and shows how some sample addresses would be rewritten by them. For more complete examples which take into account the interaction with the channel definitions, see Section 2.3.

Suppose the configuration file for the system SC.CS.EXAMPLE.COM contained the following rewrite rules shown in Example 2–1.

Example 2–1 Rewrite Rules for SC.CS.EXAMPLE.COM

```
sc                      $U@sc.cs.example.com
sc1                     $U@sc1.cs.example.com
sc2                     $U@sc2.cs.example.com
*                       $U%$&0.cs.example.com
*.cs                    $U%$&0.cs.example.com
*.cs.example            $U%$&0.cs.example.com
*.cs.example.com        $U%$&0.cs.example.com@ds.adm.example.com
sc.cs.example.com       $U@$D
sc1.cs.example.com      $U@$D
sc2.cs.example.com      $U@$D
sd.cs.example.com       $U@sd.cs.example.com
.example.com            $U%$H.example.com@cds.adm.example.com
.com                    $U@$H$D@gate.adm.example.com
[]                      $U@[$L]@gate.adm.example.com
```

Then the following initial addresses will be rewritten and routed as shown.

Initial address	Rewritten as	Routed to
user@sc	user@sc.cs.example.com	sc.cs.example.com
user@sc1	user@sc1.cs.example.com	sc1.cs.example.com
user@sc2	user@sc2.cs.example.com	sc2.cs.example.com
user@sc.cs	user@sc.cs.example.com	sc.cs.example.com
user@sc1.cs	user@sc1.cs.example.com	sc1.cs.example.com
user@sc2.cs	user@sc2.cs.example.com	sc2.cs.example.com
user@sc.cs.example	user@sc.cs.example.com	sc.cs.example.com
user@sc1.cs.example	user@sc1.cs.example.com	sc1.cs.example.com
user@sc2.cs.example	user@sc2.cs.example.com	sc2.cs.example.com
user@sc.cs.example.com	user@sc.cs.example.com	sc.cs.example.com
user@sc1.cs.example.com	user@sc1.cs.example.com	sc1.cs.example.com
user@sc2.cs.example.com	user@sc2.cs.example.com	sc2.cs.example.com
user@sd.cs.example.com	user@sd.cs.example.com	sd.cs.example.com
user@aa.cs.example.com	user@aa.cs.example.com	ds.adm.example.com
user@a.eng.example.com	user@a.eng.example.com	cds.adm.example.com
user@a.cs.example1.edu	user@a.cs.example1.edu	gate.adm.example.com — route inserted
user@b.cs.example1.edu	user@b.cs.example1.edu	gate.adm.example.com — route inserted
user@[1.2.3.4]	user@[1.2.3.4]	gate.adm.example.com — route inserted

Basically, what these rewrite rules say is: If the host name is one of our short-form names (sc, sc1 or sc2) or if it is one of our full names (sc.cs.example.com, *etc.*), expand it to our full name and route it to us. Append cs.example.com to one part shortform names

and try again. Convert one part followed by .cs to one part followed by .cs.example.com and try again. Also convert .cs.example to .cs.example.com and try again.

If the name is sd.cs.example.com (some system we connect to directly, perhaps) rewrite and route it there. If the host name is anything else in the .cs.example.com subdomain, route it to ds.cs.example.com (the gateway for the .cs.example.com subdomain). If the host name is anything else in the .example.com subdomain route it to cds.adm.example.com (the gateway for the .example.com subdomain). If the host name is anything else in the .com top-level domain route it to gate.adm.example.com (which is presumably capable of routing the message to its proper destination). If a domain literal is used send it to gate.adm.example.com as well.

Most applications of rewrite rules (like the previous example) will not change the username (or mailbox) part of the address in any way. The ability to change the username part of the address is used when PMDF is used to interface to mailers that do not conform to RFC 822 — mailers where it is necessary to stuff portions of the host/domain specification into the username part of the address. This capability should be used with great care if it is used at all.

2.2.8 Testing Domain Rewriting Rules

You can test rewrite rules with the OpenVMS command PMDF TEST/REWRITE or the UNIX or NT command `pmdf test -rewrite`. If you use a compiled configuration, then use of the /NOIMAGE qualifier (on OpenVMS) or `-noimage` qualifier (on UNIX and NT) will allow you to test changes made to the configuration file *prior* to recompiling and reinstalling the new configuration.

You may find it very instructive to rewrite a few addresses using this utility with the /DEBUG qualifier (on OpenVMS) or `-debug` qualifier (on UNIX and NT). This will show you step by step how the address is rewritten. For instance, try issuing the OpenVMS command

```
$ PMDF TEST/REWRITE/DEBUG SYSTEM@EXAMPLE.COM
```

or the UNIX command

```
# pmdf test -rewrite -debug system@example.com
```

or the NT command

```
C:\> pmdf test -rewrite -debug system@example.com
```

and see what happens.

For a description of the PMDF TEST/REWRITE (OpenVMS) or `pmdf test -rewrite` (UNIX and NT) utility, refer to Chapter 29 or Chapter 30, respectively.

2.2.9 Handling Large Numbers of Rewrite Rules

PMDF always reads in all the rewrite rules from the configuration file and stores them in memory in a hash table. Use of a compiled configuration merely bypasses the overhead associated with reading the configuration file each and every time the information is needed; a hash table is still used to store all of the rewrite rules in memory. This scheme is adequate for small to medium numbers of rewrite rules. However, some applications may require as many as 10,000 rewrite rules or more, which may consume prohibitive amounts of memory. (Applications which need many rewrite rules are channels like the ones for BITNET and UUCP: BITNET interconnects about 1,500 different systems and there are over 8,000 systems on the current UUCP map. Each of these systems needs at least one rewrite rule.)

PMDF solves this problem by providing an optional facility for storing large numbers of rewrite rules in an ancillary indexed data file. Whenever the regular configuration file is read, PMDF checks for the existence of the domain database, PMDF_DOMAIN_DATABASE.[c] If this database exists, it is opened and consulted whenever an attempted match fails on the rules found in the configuration file. The domain database is only checked if a given rule is *not* found in the configuration file, so rules can always be added to the configuration file to override those in the database.

Duplicate entries are allowed in the database only if specifically requested via CRDB/DUPLICATES (OpenVMS) or `crdb -duplicates` (UNIX and NT), at the time the database is created. (Duplicate rewrites are allowed unconditionally in the configuration file.) Entries in the database treat upper and lower case just the same way as the configuration file does; *i.e.*, patterns (left hand sides) are case insensitive, but templates (right hand sides) preserve case. Patterns in the database are limited to 32 characters and templates are limited to 80 characters unless a "long" database is built. The limits on a long database are, respectively, 80 and 256 characters.

The mere presence of the database file is enough to activate this database facility in PMDF: it is not necessary to recompile your compiled configuration. However, if you have any resident PMDF processes that need to know about this configuration change, *e.g.*, the multithreaded SMTP server, then you must restart such proceses so that they recheck PMDF configuration information and notice the new domain database. For instance, to restart the multithreaded SMTP server on OpenVMS, use the command

```
$ PMDF RESTART SMTP
```

or on UNIX, use the command

```
# pmdf restart smtp
```

The use of the domain database can be disabled with the PMDF option USE_DOMAIN_DATABASE described in Chapter 7.

The domain database should be world readable. Failure to protect the database in this fashion will make address rewriting very erratic.

[c] On OpenVMS systems this database is the file pointed at by the PMDF_DOMAIN_DATABASE logical, usually the file `PMDF_TABLE:domain.dat`; on UNIX systems this database consists of the file specified with the PMDF_DOMAIN_DATABASE option in the `/etc/pmdf_tailor` file, usually the file `/pmdf/table/domaindb.*`.

A utility is provided that can create and manipulate the domain database: CRDB (OpenVMS) or `crdb` (UNIX and NT) takes a list of rewrite rules in the same format as they appear in the configuration file and either creates a new database containing these rules or adds them to an existing database. (See Chapter 29 and Chapter 30, respectively, for full descriptions of these commands.) OpenVMS commands of the form

```
$ PMDF CRDB input-file-spec PMDF_TABLE:domain.tmp
$ RENAME PMDF_TABLE:domain.tmp PMDF_DOMAIN_DATABASE
```

or UNIX commands

```
# pmdf crdb input-file-spec PMDF_DOMAIN_DATABASE
```

or NT commands

```
C:\> pmdf crdb input-file-spec PMDF_DOMAIN_DATABASE
```

are used to read the input file, *input-file-spec*, and create a new output database. Normally, if there are rules with identical left hand sides in the input file, the first instance of such a rule will be used. CRDB or `crdb` counts such duplicates as exceptions and reports how many exceptions occurred as it exits. This behavior may be changed by specifying CRDB/DUPLICATES (OpenVMS) or `crdb -duplicates` (UNIX and NT), which causes it to create a database that allows duplicate entries.

Use the /APPEND qualifier (OpenVMS) or `-append` qualifier (UNIX and NT) to add rules to an existing domain database. When additional rules are added to an existing database in this way any duplicates will override the original rules in the existing database. CRDB or `crdb` prints a warning message when this happens.

A "long" database, as needed if the left hand sides are over 32 characters long or if the right hand sides are over 80 characters long, can be created with the OpenVMS commands

```
$ PMDF CRDB/LONG_RECORDS input-file-spec PMDF_TABLE:domain.tmp
$ RENAME PMDF_TABLE:domain.tmp PMDF_DOMAIN_DATABASE
```

or UNIX commands

```
# pmdf crdb -long_records input-file-spec PMDF_DOMAIN_DATABASE
```

or NT commands

```
C:\> pmdf crdb -long_records input-file-spec PMDF_DOMAIN_DATABASE
```

As an example, consider an input file named `bitnet.rules` whose first few lines might appear as

```
AACC              $U%$H.BITNET
AACC.BITNET       $E$F$U%$D@JNET-DAEMON
AACC.BITNET       $S$U@$D@example.com$Qtcp_local
AACC.BITNET       $U$%$D@example.com$Qtcp_local
AACC.BITNET       $U%$D@JNET-DAEMON
```

This file may be converted to a domain database with the OpenVMS commands

```
$ PMDF CRDB/DUPLICATES bitnet.rules PMDF_TABLE:domain.tmp
$ RENAME PMDF_TABLE:domain.tmp PMDF_DOMAIN_DATABASE
```

A temporary file is used so as to eliminate any window of time during which the domain database is in a mixed state or not fully generated.

2.2.10 Using Rewrites to Illegal Addresses

The ability of rewrite rules to map an address to a system that does not appear in the channel table can be used to advantage in some cases. Suppose PMDF has to contend with a strict subset of a large local DECnet network. The simplest way to handle a large local DECnet network is to place all the systems in a domain (*e.g.,* .firm.com) and use a general rewrite rule of the form:

```
.firm.com     $U%$H@decnet-mail
```

However, this does not limit access to a subset of the local DECnet; any address that ends in .firm.com will work. The alternative is to list all the accessible systems explicitly, giving each system its own entry in the rewrite rule table, and omit the more general rule. But suppose that another rewrite exists to map all non-local systems in the .com domain to a gateway system:

```
.com          $U%$H.com@gateway-system
```

Then an illegal address of the form `user@bad-system.firm.com` will be routed to the gateway, which is incorrect and may even result in a mail loop if the gateway returns the message improperly.

One solution is to list all the local systems in the channel table instead of using rewrite rules. This solves the problem at the expense of making the channel table very large. Channel table entries, unlike rewrite rules, cannot be placed in an auxiliary database. Ultimately, large numbers of channel table entries may have an adverse impact on the performance of PMDF.

A better solution is to keep the individual rewrite rules for all the local systems and insert a rule that deliberately rewrites an unknown address in the local domain to an illegal system (which might as well be the unknown system itself):

```
.firm.com     $U@$H$D
```

When an illegal local system name is used this rule will be activated before the more general .com rule is used and the address will immediately be found to be illegal.

2.2.11 Other Address Manipulations

PMDF's address rewriting facility is PMDF's primary facility for manipulating and changing the host/domain portion of addresses.[d] PMDF does, however, provide other facilities such as aliases, the address reversal database, the directory channel, and specialized mapping tables. For discussions of these facilities, refer to Chapter 3. In general, for the best performance, rewrite rules should be used whenever possible to perform address manipulations.

2.3 The Channel/host Table

PMDF consists of a large number of components, but the central unifying construct in PMDF is the channel. A channel represents a connection with another computer system or group of systems. The actual hardware connection or software transport or both may vary widely from one channel to the next. Only the system manager need know anything about PMDF's channels; users are never aware of the existence of channels and only see a single, uniform interface regardless of how messages reach their destination.

The channel/host table is stored in the PMDF configuration file; see Section 2.1 for an overview of the PMDF configuration file format.

2.3.1 Overview

Each channel consists of one or more channel programs and an outgoing message queue for storing messages that are destined to be sent to one or more of the systems associated with the channel. Channel programs perform two functions: (1) they transmit messages to remote systems, deleting them from their queue after they are sent, and (2) they accept messages from remote systems, placing them in the channel queues. Note that while a channel program only removes messages from its own queue it can enqueue messages on any queue whatsoever, including its own.

A channel program which initiates a transfer to a remote system on its own is called a "master" program, while a program which accepts transfers initiated by a remote system is called a "slave" program. A channel may be served by a master program, a slave program, or both. Either type of program may or may not be bidirectional; the direction in which a message is travelling may have nothing to do with the type of program that handles it. For example, in the case of a PhoneNet channel, the master and slave programs are both capable of transmitting and receiving messages. An SMTP channel, on the other hand, has a master program that only transmits messages and a slave program that only receives messages. These are, respectively, the SMTP client and server.

[d] Note that by using the general database and customer-supplied substitutions, it is also possible to perform complex manipulations of the username portion of an address.

2.3.2 Channel Definitions: the Channel/host Table

The second part of the PMDF configuration file gives the definitions of the channels. These definitions are collectively referred to as the "channel/host table". Each individual channel definition forms a "channel block". That is, the channel/host table defines the channels PMDF can use and the names of the systems associated with each channel. The table consists of individual blocks describing single channels. Blocks are separated by single blank lines. Comments but no blank lines may appear inside a channel block.

The first channel block in the file always describes the local channel, used to deliver messages to the local system.[e] On OpenVMS or UNIX platforms, this must be channel "l" (lowercase letter "L").

A schematic layout of a generic channel block is shown in Figure 2–1.

Figure 2–1 Channel block schematic layout

```
❶ channel-name  keyword1  keyword2 ...
❷ official-host-name  local-host-alias
❸ host-name  proper-name
```

Briefly, the two or more lines of a channel block are:

❶ The channel name followed by one or more optional keywords which alter or modify the operation of the channel. See Section 2.3.2.1 below.

❷ The official host name associated with the channel followed by an optional alias for the local host. See Section 2.3.2.2 below.

❸ Additional hosts and optional aliases for hosts reachable by the channel. This third line and subsequent lines are optional. See Section 2.3.2.3 below.

2.3.2.1 First Line: Channel Name and Keywords

The first line in a channel block gives the channel name (up to 32 characters) followed by a space and then various optional modifier keywords separated by spaces. Some keywords take arguments; see the descriptions of specific keywords for details. Channel naming conventions are discussed in Section 2.3.6; modifier keywords are described below in Section 2.3.4.

2.3.2.2 Second Line: System Name and Local Host Alias

The second line of the channel block specifies the official host name associated with the channel along with an optional alias for the local host. The line has the form:

```
official-host-name  local-host-alias
```

The official host name, *official-host-name*, should be the full name (including any subdomains or domains) of the host with which the channel communicates. In the case of the local channel, the name should be the preferred name used locally for the host PMDF

[e] The exception to this rule is the defaults channel.

is running on. In a homogeneous OpenVMS cluster environment this name will apply to the entire cluster; it does not have to be a name associated with any particular cluster node. All official host names for all channels are stored in a single common lookup table. They must be unique; duplicates are not allowed.

The local host machine is normally known by the name that appears as the official host name in the first channel block (on OpenVMS and UNIX, the l channel — note that this is a lowercase letter "L") in the configuration file. It is sometimes useful for the local host to have different names depending on the channel being used. This situation usually arises when a machine is connected to more than one network. For example, a system may need to be known as milan.uucp on the UUCP network, milan.example.com on the Internet, and milan.bitnet on BITNET.

The local host alias, *local-host-alias*, on the second line of the channel block provides this functionality. If this alias is specified, it is communicated as the local host's name to any remote hosts with which this channel communicates. This alias will replace the local host's name wherever it appears in the envelope and header of messages queued to the associated channel. If this alias is omitted the local host's official name (that is, the official host name associated with the l channel) is used.

The local host alias only affects the name of the local host. No other system names are affected. The effects of the local host alias are strictly limited to the channel to which the alias applies.

Note: The use of local host aliases is discouraged. If at all possible, each system should be known by one and only one name on all networks. Networks should strive to make this a reality. The current Internet *versus* UUCP *versus* BITNET networking fracas leads to situations where this feature is needed. In particular, it is presently impossible for a host on both BITNET and the Internet to have exactly the same name on both networks. Since different networks are associated with different channels, a per-channel local host alias is an ideal way to give the local host a per-network name.

Another Note: When a single network is involved, it may appear that local host aliases can solve lots of problems, but often the end result is a worse mess than if the proper course of action is selected — pick a single name and stick to it, living with the consequences of the conversion now instead of putting them off until it becomes even more difficult.

2.3.2.3 Additional Lines: Systems Reachable via the Channel

Any additional lines in the channel block specify additional hosts or aliases for hosts the channel can reach. These lines have the form:

```
host-name  proper-name
```

Messages to *host-name* will be queued on this channel, but To: addresses will be rewritten in the transport layer (envelope) to use *proper-name* instead of *host-name*. Addresses in the message header are **not** rewritten in this fashion. If *proper-name* is omitted the official host name for the channel will be used instead — *host-name* is then just a synonym for *official-host-name*.

All of these additional *host-name* strings are stored in the same table in which official channel host names are stored. No duplicates are allowed amongst all these names.

The functionality of these additional channel table lines may appear to duplicate some of the capabilities of rewrite rules, and this is in fact true. The ability to specify multiple hosts per channel is an older feature of PMDF that is not heavily used in more recent times. One particular usage remains, however — the mapping and unmapping of domain names to DECnet node names for hosts associated with the d channel. It is customary to use rewrite rules to canonicalize DECnet node names into full domain names and match them to the d channel. Then the channel table is used to inverse-map the domain names back into the DECnet node names. This approach results in the use of domain names in all places but envelope To: addresses, which is exactly what such systems need.

2.3.3 Envelope vs. Header Addresses: Channel-level Name Translations

Messages contain both envelope (transport layer) addresses, used by the e-mail system but generally invisible to the user, and header (display) addresses, which are the addresses visible in the message as received by the user.

PMDF's addresses always come from either the message envelope (transport layer) or the message header. Addresses can be further categorized as being either From: addresses (more generally, addresses that point back at the message source) or To: addresses (generally addresses pointing towards the message destination). PMDF does alter its address processing somewhat depending on where the address appeared.

Transport layer To: addresses are rewritten in various formats depending both on what the channel table says the channel requires. That is, channel level address rewriting may involve special forms of channel definition that request special address handling. Transport layer To: addresses (envelope To: addresses) are the only addresses where channel level rewriting is applied. Header and envelope From: addresses are not affected by channel-level translation rules. Such channel block effects on addresses, discussed above in Section 2.3.2, occur after the regular address rewriting performed by domain rewriting rules.

Thus system name transformations that should not be performed in the message header may be placed in the channel table, while transformations to be applied to the header should appear as rewrite rules. Note that in many instances, the same effect may be had using rewrite rules and the $E and $B control sequences as described in Section 2.2.6.12.

2.3.4 Channel Table Keywords

This section discusses channel keywords, which appear after the channel name on the first line of the channel definition. These keywords following the channel name are used to assign various attributes to the channel. Keywords are case insensitive, and may be up to 32 characters long; any additional characters are ignored. The supported keywords are listed alphabetically in Table 2–5 and by functional group in Table 2–6. Following the summary tables listing the keywords are sections describing each of the channel keywords in detail.

Specifying a keyword not on the list of keywords shown in Table 2–5 or Table 2–6 is not an error (although it may be incorrect). On OpenVMS systems, such undefined keywords are interpreted as rightslist identifiers, while on UNIX systems, such undefined keywords are interpreted as group ids; see Section 2.3.4.89 for more details. The PMDF TEST/REWRITE (OpenVMS) or `pmdf test -rewrite` (UNIX and NT) utility will tell you if you have any keywords in your configuration file that don't match a known rightslist identifier. See Chapter 29 or Chapter 30 for instructions on how to use PMDF TEST/REWRITE or `pmdf test -rewrite`, respectively.

Keywords shown in **bold face type** are defaults; keywords marked with † are only supported under OpenVMS; keywords marked with § are only supported for PMDF-TLS sites.

Table 2–5 Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
	`733`	2.3.4.1	Use % routing in the envelope; synonymous with `percents`
	822	2.3.4.1	Use source routes in the envelope; synonymous with `sourceroute`
	`acceptalladdresses`	2.3.4.92	Accept all recipient addresses during SMTP dialogue.
	`acceptvalidaddresses`	2.3.4.92	Accept only valid recipient addresses during SMTP dialogue.
†	`addlineaddrs`	2.3.4.94	Add all addresses from VMS MAIL TO and CC lines to PMDF headers. (Usage discouraged; use with caution.)
	`addrsperfile`	2.3.4.15	Number of addresses per message file
	`addrsperjob`	2.3.4.14	Number of addresses to be processed by a single job
	`after`	2.3.4.18	Specify time delay before master channel programs run
	`aliaslocal`	2.3.4.69	Query alias file and alias database
	`aliaspostmaster`	2.3.4.63	Redirect postmaster messages to the local channel postmaster
	`allowetrn`	2.3.4.34	Honor SMTP client ETRN commands
	`allowswitchchannel`	2.3.4.42	Allow switching to this channel from a `switchchannel` channel
	`authrewrite`	2.3.4.44	Use SMTP AUTH information in header
	`bangoverpercent`	2.3.4.2	Group A!B%C as A!(B%C)
	`bangstyle`	2.3.4.1	Use UUCP ! routing in the envelope; synonymous with `uucp`

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

Keyword	Section	Usage
bidirectional	2.3.4.7	Channel is served by both a master and slave program
blocketrn	2.3.4.34	Do not honor SMTP client ETRN commands
blocklimit	2.3.4.78	Maximum number of PMDF blocks allowed per message
cacheeverything	2.3.4.13	Cache all connection information
cachefailures	2.3.4.13	Cache only connection failure information
cachesuccesses	2.3.4.13	Cache only connection success information
channelfilter	2.3.4.86	Specify the location of channel filter file; synonym for destinationfilter
charset7	2.3.4.51	Default character set to associate with 7-bit text messages
charset8	2.3.4.51	Default character set to associate with 8-bit text messages
charsetesc	2.3.4.51	Default character set to associate with text containing the escape character
checkehlo	2.3.4.32	Check the SMTP response banner for whether to use EHLO
client_auth	2.3.4.43	Specify which CLIENT_AUTH section to use for client SASL
commentinc	2.3.4.67	Leave comments in message header lines intact
commentomit	2.3.4.67	Remove comments from message header lines
commentstrip	2.3.4.67	Remove problematic characters from comment field in message header lines
commenttotal	2.3.4.67	Strip comments (material in parentheses) everywhere
connectalias	2.3.4.5	Do not rewrite addresses upon message dequeue
connectcanonical	2.3.4.5	Rewrite addresses upon message dequeue
convert_octet_stream	2.3.4.54	Convert application/octet-stream material as appropriate
copysendpost	2.3.4.21	Send copies of failures to the postmaster unless the originator address is blank
copywarnpost	2.3.4.22	Send copies of warnings to the postmaster unless the originator address is blank
daemon	2.3.4.81	Specify name of a gateway daemon (host) to route to
datefour	2.3.4.72	Convert date/time specifications to four digit years
datetwo	2.3.4.72	Convert date/time specifications to two digit years
dayofweek	2.3.4.73	Include day of week in date/time specifications
defaulthost	2.3.4.47	Specify a domain name to use to complete addresses
defaultmx	2.3.4.38	Channel determines whether or not to do MX lookups from network
defaultnameservers	2.3.4.38	Consult TCP/IP stack's choice of nameservers
deferred	2.3.4.19	Honor deferred delivery dates
defragment	2.3.4.76	Reassemble any MIME-compliant message/partial parts queued to this channel
description	2.3.4.87	Channel description
destinationfilter	2.3.4.86	Specify the location of channel filter file to apply to outgoing messages
disableetrn	2.3.4.34	Disable support for the ETRN SMTP command

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
	domainetrn	2.3.4.34	Honor only those SMTP client ETRN commands that specify a domain
	domainvrfy	2.3.4.35	Issue SMTP VRFY commands using full address
	dropblank	2.3.4.49	Strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
	ehlo	2.3.4.32	Use EHLO on all initial SMTP connections
	eightbit	2.3.4.50	Channel supports eight bit characters
	eightnegotiate	2.3.4.50	Channel should negotiate use of eight bit transmission if possible
	eightstrict	2.3.4.50	Channel should reject messages that contain unnegotiated eight bit data
	errsendpost	2.3.4.21	Send copies of failures to the postmaster if the originator address is illegal
	errwarnpost	2.3.4.22	Send copies of warnings to the postmaster if the originator address is illegal
	expandchannel	2.3.4.16	Channel in which to perform deferred expansion due to application of expandlimit
	expandlimit	2.3.4.16	Process an incoming message "off-line" when the number of addressees exceeds this limit
	exproute	2.3.4.3	Explicit routing for this channel's addresses
	exquota	2.3.4.80	On OpenVMS, use EXQUOTA privileges if necessary to deliver VMS MAIL messages; on UNIX treat as holdexquota for Berkeley mailboxes; on all platforms deliver to overquota PMDF MessageStore or PMDF popstore accounts
	fileinto	2.3.4.86	Specify effect on address when a mailbox filter fileinto operation is applied
	filesperjob	2.3.4.14	Number of queue entries to be processed by a single job
	filter	2.3.4.86	Specify the location of user filter files
†	foreign	2.3.4.53	Use VMS MAIL's foreign message format as needed with VMS MAIL
	forwardcheckdelete	2.3.4.40	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, delete the name and use the IP address
	forwardchecknone	2.3.4.40	Do not perform a forward lookup after a DNS reverse lookup
	forwardchecktag	2.3.4.40	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, tag the name with *
†	goldmail	2.3.4.26	Generate Gold-Mail compatible read receipts
	grey	2.3.4.83	Use Grey Book address formats (inverted order domains)
	header_733	2.3.4.1	Use % routing in the message header
	header_822	2.3.4.1	Use source routes in the message header
	header_uucp	2.3.4.1	Use ! routing in the header

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
†	headerbottom	2.3.4.58	Place the message header at the bottom of the message (usage discouraged; use with caution; see 2.3.4.58)
	headerinc	2.3.4.58	Place the message header at the top of the message
	headerlabelalign	2.3.4.75	Align headers
	headerlinelength	2.3.4.75	Fold long headers
†	headeromit	2.3.4.58	Omit the message header from the message (usage discouraged; use with caution; see 2.3.4.58)
	headerread	2.3.4.59	Apply source channel header trimming rules from an options file to the message headers before headers are processed (use with caution)
	headertrim	2.3.4.59	Apply destination channel header trimming rules from an options file to the message headers after headers are processed (use with caution)
	holdexquota	2.3.4.80	Hold messages for users that are over quota
	holdlimit	2.3.4.16	.HELD an incoming message when the number of addressees exceeds this limit
	identnone	2.3.4.40	Perform IP to hostname translation; include both hostname and IP address in Received: header
	identnonelimited	2.3.4.40	Perform IP to hostname translation, but do not use the hostname during channel switching; include both hostname and IP address in Received: header
	identnonenumeric	2.3.4.40	Do not perform IP to hostname translation
	identnonesymbolic	2.3.4.40	Perform IP to hostname translation; include only the hostname in Received: header
	ignoreencoding	2.3.4.60	Ignore Encoding: header on incoming messages
	ignoremessageencoding	2.3.4.60	Ignore Encoding: header in embedded messages
	ignoremultipartencoding	2.3.4.60	Ignore Encoding: header in multipart messages
	immediate	2.3.4.9	Delivery started immediately after submission for messages of second-class or higher priority
	immnonurgent	2.3.4.9	Delivery started immediately after submission even for messages with lower than normal priority
	immnormal	2.3.4.9	Delivery started immediately after submission for messages of normal or higher priority
	immurgent	2.3.4.9	Delivery started immediately after submission for urgent messages only
	improute	2.3.4.3	Implicit routing for this channel's addresses
	includefinal	2.3.4.27	Include final form of address in delivery notifications
	inline	2.3.4.90	Perform directory lookups immediately
	inner	2.3.4.56	Rewrite inner message headers
	innertrim	2.3.4.59	Apply header trimming rules from an options file to inner message headers (use with caution)
	interfaceaddress	2.3.4.37	Bind to the specified TCP/IP interface address
	interpretencoding	2.3.4.60	Interpret Encoding: header on incoming messages
	interpretmessageencoding	2.3.4.60	Interpret Encoding: header in embedded messages
	interpretmultipartencoding	2.3.4.60	Interpret Encoding: header in multipart messages
	lastresort	2.3.4.39	Specify a last resort host
	linelength	2.3.4.52	Message lines exceeding this length limit will be wrapped
	linelimit	2.3.4.78	Maximum number of lines allowed per message

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
	localvrfy	2.3.4.35	Issue SMTP VRFY command using local address
	logging	2.3.4.84	Log message enqueues and dequeues into the log file
†	logicaldisk	2.3.4.30	Spread PMDF channel queues across multiple disks
	loopcheck	2.3.4.91	Automatically detect mail loops when sending.
	mailfromdnsverify	2.3.4.41	Verify that the domain specified on MAIL FROM: line is in the DNS
	master	2.3.4.7	Channel is served only by a master program
	master_debug	2.3.4.85	Generate debugging output in the channel's master program output
	maxblocks	2.3.4.77	Maximum number of PMDF blocks per message; longer messages are broken into multiple messages
	maxheaderaddrs	2.3.4.74	Maximum number of addresses per message header line; longer header lines are broken into multiple header lines
	maxheaderchars	2.3.4.74	Maximum number of characters per message header line; longer header lines are broken into multiple header lines
	maxjobs	2.3.4.14	Maximum number of jobs which can be created at once
	maxlines	2.3.4.77	Maximum number of message lines per message; longer messages are broken into multiple messages
	maxperiodicnonurgent	2.3.4.11	Specify that periodic jobs should only process messages of nonurgent or lower priority
	maxperiodicnormal	2.3.4.11	Specify that periodic jobs should only process messages of normal or lower priority
	maxperiodicurgent	2.3.4.11	Specify that periodic jobs should process messages of urgent or lower priority
	maxprocchars	2.3.4.79	Specify maximum length of headers to process
	maysasl	2.3.4.43	Allow SMTP server and client SASL authentication
	maysaslclient	2.3.4.43	SMTP client attempts to use SASL authentication
	maysaslserver	2.3.4.43	SMTP server offers SASL authentication
§	maytls	2.3.4.45	SMTP client and server allow TLS use
§	maytlsclient	2.3.4.45	SMTP client will attempt TLS use
§	maytlsserver	2.3.4.45	SMTP server allows TLS use
	minperiodicnonurgent	2.3.4.11	Specify that periodic jobs should only process messages of nonurgent or higher priority
	minperiodicnormal	2.3.4.11	Specify that periodic jobs should only process messages of normal or higher priority
	minperiodicurgent	2.3.4.11	Specify that periodic jobs should only process messages of urgent priority
	missingrecipientpolicy	2.3.4.48	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
	msexchange	2.3.4.46	Channel serves MS Exchange gateways
	multigate	2.3.4.82	Channel serves multiple BITNET gateways
	multiple	2.3.4.15	Accepts multiple destination hosts in a single message copy
	mustsasl	2.3.4.43	Must use SASL authentication
	mustsaslclient	2.3.4.43	SMTP client insists upon SASL authentication
	mustsaslserver	2.3.4.43	SMTP server insists upon SASL authentication

§Supported only for PMDF-TLS sites.

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
§	musttls	2.3.4.45	SMTP client and server insist upon TLS use and will not transfer messages with remote sides that do not support TLS
§	musttlsclient	2.3.4.45	SMTP client insists upon TLS use and will not send messages to any remote SMTP server that does not support TLS use
§	musttlsserver	2.3.4.45	SMTP server insists upon TLS use and will not accept messages from any remote SMTP client that does not support TLS use
	mx	2.3.4.38	TCP/IP network and software supports MX record lookups
	nameservers	2.3.4.38	Consult specified nameservers rather than TCP/IP stack's choice
†	network	2.3.4.89	NETMBX privilege required for use
†	**noaddlineaddrs**	2.3.4.94	Only addresses processed by PMDF are included in headers for mail sent from VMS MAIL. (default)
	nobangoverpercent	2.3.4.2	Group A!B%C as (A!B)%C (default)
	noblocklimit	2.3.4.78	No limit specified for the number of PMDF blocks allowed per message
	nocache	2.3.4.13	Do not cache any connection information
	nochannelfilter	2.3.4.86	Do not perform channel filtering for outgoing messages; synonym for nodestinationfilter
	noconvert_octet_stream	2.3.4.54	Do not convert application/octet-stream material
	nodayofweek	2.3.4.73	Remove day of week from date/time specifications
	nodefaulthost	2.3.4.47	Do not specify a domain name to use to complete addresses
	nodeferred	2.3.4.19	Do not honor deferred delivery dates
	nodefragment	2.3.4.76	Do not perform special processing for message/partial messages
	nodestinationfilter	2.3.4.86	Do not perform channel filtering for outgoing messages
†	nodns	2.3.4.38	TCP/IP network does not support DNS (nameserver) lookups
	nodropblank	2.3.4.49	Do not strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
	noehlo	2.3.4.32	Never use the SMTP EHLO command
	noexproute	2.3.4.3	No explicit routing for this channel's addresses
	noexquota	2.3.4.80	Return to originator any messages to users who are over quota
	nofileinto	2.3.4.86	Mailbox filter fileinto operator has no effect
	nofilter	2.3.4.86	Do not perform user mailbox filtering
†	**noforeign**	2.3.4.53	Do not use VMS MAIL's foreign message format
†	**nogoldmail**	2.3.4.26	Do not generate Gold-Mail compatible read receipts
	nogrey	2.3.4.83	Do not use Grey Book address formats
	noheaderread	2.3.4.59	Do not apply header trimming rules from option file upon message enqueue
	noheadertrim	2.3.4.59	Do not apply header trimming rules from options file
	noimproute	2.3.4.3	No implicit routing for this channel's addresses
	noinline	2.3.4.90	Do not do directory channel lookups immediately
	noinner	2.3.4.56	Do not rewrite inner message headers

§Supported only for PMDF-TLS sites.

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
	noinnertrim	2.3.4.59	Do not apply header trimming to inner message headers
	nolinelimit	2.3.4.78	No limit specified for the number of lines allowed per message
	nologging	2.3.4.84	Do not log message enqueues and dequeues into the log file
†	**nologicaldisk**	2.3.4.30	Store PMDF channel queues on a single disk
	nomailfromdnsverify	2.3.4.41	Do not perform DNS domain verification on the MAIL FROM: address
	nomaster_debug	2.3.4.85	Do not generate debugging output in the channel's master program output
	nomsexchange	2.3.4.46	Channel does not serve MS Exchange gateways
	nomultigate	2.3.4.82	Channel does not serve multiple BITNET gateways
	nomx	2.3.4.38	TCP/IP network does not support MX lookups
	nonrandommx	2.3.4.38	Do MX lookups; do not randomize returned entries with equal precedence
	nonurgentblocklimit	2.3.4.10	Force messages above this size to wait unconditionally for a periodic job
	nonurgentnotices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned for messages of non-urgent priority
†	nonurgentqueue	2.3.4.18	Specify the queue for master channel program processing of nonurgent messages
	noreceivedfor	2.3.4.62	Do not include envelope to address in Received: header
	noreceivedfrom	2.3.4.62	Do not include the envelope From: address when constructing Received: header
	norelaxheadertermination	2.3.4.93	Don't consider a line with just spaces and tabs to be a header terminator.
	noremotehost	2.3.4.47	Use local host's domain name as the default domain name to complete addresses
	norestricted	2.3.4.57	Do not apply RFC 1137 restricted encoding to addresses
	noreturnaddress	2.3.4.63	Use the RETURN_ADDRESS option value
	noreturnpersonal	2.3.4.63	Use the RETURN_PERSONAL option value
	noreverse	2.3.4.55	Do not apply reverse database to addresses
	normalblocklimit	2.3.4.10	Force messages above this size to nonurgent priority
	normalnotices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned for messages of normal priority
†	normalqueue	2.3.4.18	Specify the queue for master channel program processing of normal messages
	norules	2.3.4.6	Do not do channel-specific rewrite rule checks
	nosasl	2.3.4.43	SASL authentication not attempted or permitted
	nosaslclient	2.3.4.43	SMTP client does not attempt SASL authentication
	nosaslserver	2.3.4.43	SMTP server does not permit SASL authentication
	nosaslswitchchannel	2.3.4.43	Do not allow switching to this channel upon successful SASL authentication
	nosendetrn	2.3.4.33	Do not send SMTP ETRN command
	nosendpost	2.3.4.21	Do not send copies of failures to the postmaster

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
	noserviceall	2.3.4.12	Immediate delivery jobs process only the messages they were queued to process
	noslave_debug	2.3.4.85	Do not generate debugging output in the channel's slave program output
	nosmtp	2.3.4.31	Channel does not use SMTP
	nosourcefilter	2.3.4.86	Do not perform channel filtering for incoming messages
	noswitchchannel	2.3.4.42	Stay with the server channel; do not switch to the channel associated with the originating host; do not permit being switched to
	notices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned
§	**notls**	2.3.4.45	SMTP client and server neither attempt nor allow TLS use
§	notlsclient	2.3.4.45	SMTP client does not attempt TLS use when sending messages
§	notlsserver	2.3.4.45	SMTP server does not offer or allow TLS use when receiving messages
	novrfy	2.3.4.35	Do not issue SMTP VRFY commands
	nowarnpost	2.3.4.22	Do not send copies of warnings to the postmaster
	nox_env_to	2.3.4.61	Do not add X-Envelope-to: header lines while enqueuing
	percents	2.3.4.1	Use % routing in the envelope; synonymous with 733
	period	2.3.4.9	Specify periodicity of periodic channel service
	periodic	2.3.4.9	Channel is serviced only periodically; immediate delivery processing is never done
	personalinc	2.3.4.68	Leave personal names in message header lines intact
	personalomit	2.3.4.68	Remove personal name fields from message header lines
	personalstrip	2.3.4.68	Strip problematic characters from personal name fields in message header lines
	port	2.3.4.37	Send to the specified TCP/IP port
	postheadbody	2.3.4.23	Both the message's header and body are sent to the postmaster when a delivery failure occurs
	postheadonly	2.3.4.23	Only the message's header is sent to the postmaster when a delivery failure occurs
	queue	2.3.4.18	Specify queue master channel programs run in
	randommx	2.3.4.38	Do MX lookups; randomize returned entries with equal precedence
	readreceiptmail	2.3.4.25	Ignore read receipt requests when delivering to VMS MAIL, rather than "downgrading" them to delivery receipt requests; leave it up to user agents to act upon the read receipt request
	receivedfor	2.3.4.62	Include envelope to address in Received: header
	receivedfrom	2.3.4.62	Include the envelope From: address when constructing Received: header
	relaxheadertermination	2.3.4.93	Consider a line with just spaces and tabs to be a header terminator.

§Supported only for PMDF-TLS sites.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

Keyword	Section	Usage
remotehost	2.3.4.47	Use remote host's name as the default domain name to complete addresses
reportboth	2.3.4.24	Generate both header and NOTARY delivery receipt requests from "foreign" delivery receipt requests
reportheader	2.3.4.24	Generate only header delivery receipt requests from "foreign" delivery receipt requests
reportnotary	2.3.4.24	Generate only NOTARY delivery receipt requests from "foreign" delivery receipt requests
reportsuppress	2.3.4.24	Suppress delivery receipt requests from "foreign" delivery receipt requests
restricted	2.3.4.57	Apply RFC 1137 restricted encoding to addresses
returnaddress	2.3.4.63	Set the return address for the local Postmaster
returnenvelope	2.3.4.64	Control use of blank envelope return addresses
returnpersonal	2.3.4.63	Set the personal name for the local Postmaster
reverse	2.3.4.55	Apply reverse database or REVERSE mapping to addresses
routelocal	2.3.4.4	Rewriting should shortcircuit routing addresses
rules	2.3.4.6	Do channel-specific rewrite rule checks
saslswitchchannel	2.3.4.43	Switch to another channel when SASL authentication is successful
sendetrn	2.3.4.33	Send SMTP ETRN command
sendpost	2.3.4.21	Send copies of failures to the postmaster
sensitivitycompanyconfidential	2.3.4.88	Allow messages of any sensitivity
sensitivitynormal	2.3.4.88	Reject messages whose sensitivity is higher than normal
sensitivitypersonal	2.3.4.88	Reject messages whose sensitivity is higher than personal
sensitivityprivate	2.3.4.88	Reject messages whose sensitivity is higher than private
serviceall	2.3.4.12	Immediate delivery jobs process all messages (queued for the channel)
sevenbit	2.3.4.50	Channel does not support eight bit characters; eight bit characters must be encoded
silentetrn	2.3.4.34	Honor SMTP client ETRN commands, without echoing channel information
single	2.3.4.15	Only one envelope To: address per message copy
single_sys	2.3.4.15	Each message copy must be for a single destination system
slave	2.3.4.7	Channel is serviced only by a slave program
slave_debug	2.3.4.85	Generate debugging output in the channel's slave program output
smtp	2.3.4.31	Channel uses SMTP
smtp_cr	2.3.4.31	Accept CR as an SMTP line terminator
smtp_crlf	2.3.4.31	Require CRLF as the SMTP line terminator
smtp_crorlf	2.3.4.31	Allow any of CR, LF, or CRLF as the SMTP line terminator
smtp_lf	2.3.4.31	Accept LF as an SMTP line terminator
sourceblocklimit	2.3.4.78	Maximum number of PMDF blocks allowed per incoming message
sourcecommentinc	2.3.4.67	Leave comments in incoming message header lines intact

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

	Keyword	Section	Usage
	sourcecommentomit	2.3.4.67	Remove comments from incoming message header lines
	sourcecommentstrip	2.3.4.67	Remove problematic characters from comment field in incoming message header lines
	sourcecommenttotal	2.3.4.67	Strip comments (material in parentheses) everywhere in incoming messages
	sourcefilter	2.3.4.86	Specify the location of channel filter file for incoming messages
	sourcepersonalinc	2.3.4.68	Leave personal names in incoming message header lines intact
	sourcepersonalomit	2.3.4.68	Remove personal name fields from incoming message header lines
	sourcepersonalstrip	2.3.4.68	Strip problematic characters from personal name fields in incoming message header lines
	sourceroute	2.3.4.1	Use source routes in the message envelope; synonymous with 822
	streaming	2.3.4.28	Specify degree of protocol streaming for channel to use
	subaddressexact	2.3.4.71	Alias must match exactly, including exact subaddress match
	subaddressrelaxed	2.3.4.71	Alias without subaddress may match
	subaddresswild	2.3.4.71	Alias with subaddress wildcard may match
	subdirs	2.3.4.17	Use multiple subdirectories
	submit	2.3.4.8	Mark the channel as a submit-only channel
	suppressfinal	2.3.4.27	Include only original form of address in notification messages
	switchchannel	2.3.4.42	Switch from the server channel to the channel associated with the originating host
	threaddepth	2.3.4.29	Number of messages triggering new thread with multithreaded SMTP client
§	tlsswitchchannel	2.3.4.45	Switch to specified channel upon successful TLS negotiation
	unrestricted	2.3.4.57	Do not apply RFC 1137 restricted encoding to addresses
	urgentblocklimit	2.3.4.10	Force messages above this size to normal priority
	urgentnotices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority
†	urgentqueue	2.3.4.18	Specify the queue for master channel program processing of urgent messages
	user	<REFERENCE TYPE="HEAD2" CHANNELORIGINATORTAG\VALUE)	Specify account under which to run the pipe channel or specify Message Router mailbox name
	usereplyto	2.3.4.65	Specify mapping of Reply-to: header
	useresent	2.3.4.66	Specify mapping of Resent- headers for non RFC 822 environments
	uucp	2.3.4.1	Use UUCP ! routing in the envelope; synonymous with bangstyle
	validatelocalmsgstore	2.3.4.70	Enqueuing channels check that the local part of addresses they enqueue to this channel matches a PMDF Message Store account

§Supported only for PMDF-TLS sites.

†Supported only on OpenVMS.

Table 2–5 (Cont.) Channel Block Keywords Listed Alphabetically

Keyword	Section	Usage
validatelocalnone	2.3.4.70	Enqueuing channels perform no validatation check on the local part of addresses they enqueue to this channel
validatelocalsystem	2.3.4.70	Enqueuing channels check that the local part of addresses they enqueue to this channel matches an account on the system
vrfyallow	2.3.4.36	Provide informative responses to SMTP VRFY command
vrfydefault	2.3.4.36	Default responses to SMTP VRFY command, according to channel's HIDE_VERIFY option setting
vrfyhide	2.3.4.36	Provide obfuscatory responses to SMTP VRFY command
warnpost	2.3.4.22	Send copies of warnings to the postmaster
x_env_to	2.3.4.61	Add X-Envelope-to: header lines while enqueuing

Table 2–5 above lists channel keywords alphabetically; Table 2–6 below lists channel keywords by functional group. Keywords shown in **bold face type** are defaults; keywords marked with † are only supported under OpenVMS; keywords marked with § are only supported for PMDF-TLS sites.

Table 2–6 Channel Block Keywords Grouped by Functionality

Keyword	Section	Usage
	Addresses	
733	2.3.4.1	Use % routing in the envelope; synonymous with percents
822	2.3.4.1	Use source routes in the envelope; synonymous with sourceroute
acceptalladdresses	2.3.4.92	Accept all recipient addresses during SMTP dialogue.
acceptvalidaddresses	2.3.4.92	Accept only valid recipient addresses during SMTP dialogue.
aliaslocal	2.3.4.69	Query alias file and alias database
authrewrite	2.3.4.44	Use SMTP AUTH information in header
bangoverpercent	2.3.4.2	Group A!B%C as A!(B%C)
bangstyle	2.3.4.1	Use UUCP ! routing in the envelope; synonymous with uucp
defaulthost	2.3.4.47	Specify a domain name to use to complete addresses
exproute	2.3.4.3	Explicit routing for this channel's addresses
grey	2.3.4.83	Use Grey Book address formats (inverted order domains)
holdlimit	2.3.4.16	.HELD an incoming message when the number of addressees exceeds this limit
improute	2.3.4.3	Implicit routing for this channel's addresses
inline	2.3.4.90	Perform directory channel lookups immediately
missingrecipientpolicy	2.3.4.48	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
nobangoverpercent	2.3.4.2	Group A!B%C as (A!B)%C (default)

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		Addresses	
	nodefaulthost	2.3.4.47	Do not specify a domain name to use to complete addresses
	noexproute	2.3.4.3	No explicit routing for this channel's addresses
	nogrey	2.3.4.83	Do not use Grey Book address formats
	noimproute	2.3.4.3	No implicit routing for this channel's addresses
	noinline	2.3.4.90	Do not do directory channel lookups immediately
	noremotehost	2.3.4.47	Use local host's domain name as the default domain name to complete addresses
	norestricted	2.3.4.57	Do not apply RFC 1137 restricted encoding to addresses
	noreverse	2.3.4.55	Do not apply reverse database to addresses
	norules	2.3.4.6	Do not do channel-specific rewrite rule checks
	percents	2.3.4.1	Use % routing in the envelope; synonymous with 733
	remotehost	2.3.4.47	Use remote host's name as the default domain name to complete addresses
	restricted	2.3.4.57	Apply RFC 1137 restricted encoding to addresses
	reverse	2.3.4.55	Apply reverse database or REVERSE mapping to addresses
	routelocal	2.3.4.4	Rewriting should shortcircuit routing addresses
	rules	2.3.4.6	Do channel-specific rewrite rule checks
	sourceroute	2.3.4.1	Use source routes in the message envelope; synonymous with 822
	subaddressexact	2.3.4.71	Alias must match exactly, including exact subaddress match
	subaddressrelaxed	2.3.4.71	Alias without subaddress may match
	subaddresswild	2.3.4.71	Alias with subaddress wildcard may match
	unrestricted	2.3.4.57	Do not apply RFC 1137 restricted encoding to addresses
	uucp	2.3.4.1	Use UUCP ! routing in the envelope; synonymous with bangstyle
	validatelocalmsgstore	2.3.4.70	Enqueuing channels check that the local part of addresses they enqueue to this channel matches a PMDF Message Store account
	validatelocalnone	2.3.4.70	Enqueuing channels perform no validatation check on the local part of addresses they enqueue to this channel
	validatelocalsystem	2.3.4.70	Enqueuing channels check that the local part of addresses they enqueue to this channel matches an account on the system
		Attachments and MIME processing	
	convert_octet_stream	2.3.4.54	Convert application/octet-stream material as appropriate
	defragment	2.3.4.76	Reassemble any MIME-compliant message/partial parts queued to this channel
†	foreign	2.3.4.53	Use VMS MAIL's foreign message format as needed with VMS MAIL
	ignoreencoding	2.3.4.60	Ignore Encoding: header on incoming messages

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
	Attachments and MIME processing		
	ignoremessageencoding	2.3.4.60	Ignore Encoding: header on embedded messages
	ignoremultipartencoding	2.3.4.60	Ignore Encoding: header on multipart messages
	interpretencoding	2.3.4.60	Interpret Encoding: header on incoming messages
	interpretmessageencoding	2.3.4.60	Interpret Encoding: header on embedded messages
	interpretmultipartencoding	2.3.4.60	Interpret Encoding: header on multipart messages
	linelength	2.3.4.52	Message lines exceeding this length limit will be wrapped
	maxblocks	2.3.4.77	Maximum number of PMDF blocks per message; longer messages are broken into multiple messages
	maxlines	2.3.4.77	Maximum number of message lines per message; longer messages are broken into multiple messages
	noconvert_octet_stream	2.3.4.54	Do not convert application/octet-stream material
	nodefragment	2.3.4.76	Do not perform special processing for message/partial messages
†	**noforeign**	2.3.4.53	Do not use VMS MAIL's foreign message format
	nolinelimit	2.3.4.78	No limit specified for the number of lines allowed per message
	Character sets and eight bit data		
	charset7	2.3.4.51	Default character set to associate with 7-bit text messages
	charset8	2.3.4.51	Default character set to associate with 8-bit text messages
	charsetesc	2.3.4.51	Default character set to associate with text containing the escape character
	eightbit	2.3.4.50	Channel supports eight bit characters
	eightnegotiate	2.3.4.50	Channel should negotiate use of eight bit transmission if possible
	eightstrict	2.3.4.50	Channel should reject messages that contain unnegotiated eight bit data
	sevenbit	2.3.4.50	Channel does not support eight bit characters; eight bit characters must be encoded
	File creation in the PMDF queue area		
	addrsperfile	2.3.4.15	Number of addresses per message file
	expandchannel	2.3.4.16	Channel in which to perform deferred expansion due to application of expandlimit
	expandlimit	2.3.4.16	Process an incoming message "off-line" when the number of addressees exceeds this limit
†	logicaldisk	2.3.4.30	Spread PMDF channel queues across multiple disks
	multiple	2.3.4.15	Accepts multiple destination hosts in a single message copy
†	**nologicaldisk**	2.3.4.30	Store PMDF channel queues on a single disk
	single	2.3.4.15	Only one envelope To: address per message copy
	single_sys	2.3.4.15	Each message copy must be for a single destination system
	subdirs	2.3.4.17	Use multiple subdirectories

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		Gateway/firewall/mailhub/Message Router channel connection	
	daemon	2.3.4.81	Specify name of a gateway daemon (host) to route to
	lastresort	2.3.4.39	Specify a last resort host
	multigate	2.3.4.82	Channel serves multiple BITNET gateways
	nomultigate	2.3.4.82	Channel does not serve multiple BITNET gateways
	user	<REFERENCE.IHEAD2 CHANNELORIGINATORTAG VAR	Specify account under which to run the pipe channel or specify Message Router mailbox name
		Headers	
†	addlineaddrs	2.3.4.94	Add all addresses from VMS MAIL TO and CC lines to PMDF headers. (Usage discouraged; use with caution.)
	authrewrite	2.3.4.44	Use SMTP AUTH information in header
	commentinc	2.3.4.67	Leave comments in message header lines intact
	commentomit	2.3.4.67	Remove comments from message header lines
	commentstrip	2.3.4.67	Remove problematic characters from comment field in message header lines
	commenttotal	2.3.4.67	Strip comments (material in parentheses) everywhere
	datefour	2.3.4.72	Convert date/time specifications to four digit years
	datetwo	2.3.4.72	Convert date/time specifications to two digit years
	dayofweek	2.3.4.73	Include day of week in date/time specifications
	defaulthost	2.3.4.47	Specify a domain name to use to complete addresses
	dropblank	2.3.4.49	Strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
	header_733	2.3.4.1	Use % routing in the message header
	header_822	2.3.4.1	Use source routes in the message header
	header_uucp	2.3.4.1	Use ! routing in the header
†	headerbottom	2.3.4.58	Place the message header at the bottom of the message (usage discouraged; use with caution; see 2.3.4.58)
	headerinc	2.3.4.58	Place the message header at the top of the message
	headerlabelalign	2.3.4.75	Align headers
	headerlinelength	2.3.4.75	Fold long headers
†	headeromit	2.3.4.58	Omit the message header from the message (usage discouraged; use with caution; see 2.3.4.58)
	headerread	2.3.4.59	Apply source channel header trimming rules from an options file to the message headers before headers are processed (use with caution)
	headertrim	2.3.4.59	Apply destination channel header trimming rules from an options file to the message headers after headers are processed (use with caution)
	inner	2.3.4.56	Rewrite inner message headers
	innertrim	2.3.4.59	Apply header trimming rules from an options file to inner message headers (use with caution)

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		Headers	
	maxheaderaddrs	2.3.4.74	Maximum number of addresses per message header line; longer header lines are broken into multiple header lines
	maxheaderchars	2.3.4.74	Maximum number of characters per message header line; longer header lines are broken into multiple header lines
	missingrecipientpolicy	2.3.4.48	Set policy for how to legalize (which header to add) messages that are lacking any recipient headers
†	**noaddlineaddrs**	2.3.4.94	Only addresses processed by PMDF are included in headers for mail sent from VMS MAIL. (default)
	nodayofweek	2.3.4.73	Remove day of week from date/time specifications
	nodefaulthost	2.3.4.47	Do not specify a domain name to use to complete addresses
	nodropblank	2.3.4.49	Do not strip blank To:, Resent-To:, Cc:, or Resent-Cc: headers
	noheaderread	2.3.4.59	Do not apply header trimming rules from option file upon message enqueue
	noheadertrim	2.3.4.59	Do not apply header trimming rules from options file
	noinner	2.3.4.56	Do not rewrite inner message headers
	noinnertrim	2.3.4.59	Do not apply header trimming to inner message headers
	noreceivedfor	2.3.4.62	Do not include envelope to address in Received: header
	noreceivedfrom	2.3.4.62	Do not include the envelope From: address when constructing Received: header
	norelaxheadertermination	2.3.4.93	Don't consider a line with just spaces and tabs to be a header terminator.
	noremotehost	2.3.4.47	Use local host's domain name as the default domain name to complete addresses
	norestricted	2.3.4.57	Do not apply RFC 1137 restricted encoding to addresses
	noreverse	2.3.4.55	Do not apply reverse database to addresses
	norules	2.3.4.6	Do not do channel-specific rewrite rule checks
	nox_env_to	2.3.4.61	Do not add X-Envelope-to: header lines while enqueuing
	personalinc	2.3.4.68	Leave personal names in message header lines intact
	personalomit	2.3.4.68	Remove personal name fields from message header lines
	personalstrip	2.3.4.68	Strip problematic characters from personal name fields in message header lines
	receivedfor	2.3.4.62	Include envelope to address in Received: header
	receivedfrom	2.3.4.62	Include the envelope From: address when constructing Received: header
	relaxheadertermination	2.3.4.93	Consider a line with just spaces and tabs to be a header terminator.
	remotehost	2.3.4.47	Use remote host's name as the default domain name to complete addresses
	restricted	2.3.4.57	Apply RFC 1137 restricted encoding to addresses

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		Headers	
	reverse	2.3.4.55	Apply reverse database or REVERSE mapping to addresses
	rules	2.3.4.6	Do channel-specific rewrite rule checks
	sensitivitycompanyconfidential	2.3.4.88	Allow messages of any sensitivity
	sensitivitynormal	2.3.4.88	Reject messages whose sensitivity is higher than normal
	sensitivitypersonal	2.3.4.88	Reject messages whose sensitivity is higher than personal
	sensitivityprivate	2.3.4.88	Reject messages whose sensitivity is higher than private
	sourcecommentinc	2.3.4.67	Leave comments in incoming message header lines intact
	sourcecommentomit	2.3.4.67	Remove comments from incoming message header lines
	sourcecommentstrip	2.3.4.67	Remove problematic characters from comment field in incoming message header lines
	sourcecommenttotal	2.3.4.67	Strip comments (material in parentheses) everywhere in incoming messages
	sourcepersonalinc	2.3.4.68	Leave personal names in incoming message header lines intact
	sourcepersonalomit	2.3.4.68	Remove personal name fields from incoming message header lines
	sourcepersonalstrip	2.3.4.68	Strip problematic characters from personal name fields in incoming message header lines
	unrestricted	2.3.4.57	Do not apply RFC 1137 restricted encoding to addresses
	usereplyto	2.3.4.65	Specify mapping of Reply-to: header
	useresent	2.3.4.66	Specify mapping of Resent- headers for non RFC 822 environments
	x_env_to	2.3.4.61	Add X-Envelope-to: header lines while enqueuing
		Incoming channel matching and switching	
	allowswitchchannel	2.3.4.42	Allow switching to this channel from a switchchannel channel
	nosaslswitchchannel	2.3.4.43	Do not allow switching to this channel upon successful SASL authentication
	noswitchchannel	2.3.4.42	Stay with the server channel; do not switch to the channel associated with the originating host; do not permit being switched to
	saslswitchchannel	2.3.4.43	Switch to another channel when SASL authentication is successful
	switchchannel	2.3.4.42	Switch from the server channel to the channel associated with the originating host
§	tlsswitchchannel	2.3.4.45	Switch to specified channel upon successful TLS negotiation
		Logging and debugging	

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

Keyword	Section	Usage
	Logging and debugging	
logging	2.3.4.84	Log message enqueues and dequeues into the log file
master_debug	2.3.4.85	Generate debugging output in the channel's master program output
nologging	2.3.4.84	Do not log message enqueues and dequeues into the log file
nomaster_debug	2.3.4.85	Do not generate debugging output in the channel's master program output
noslave_debug	2.3.4.85	Do not generate debugging output in the channel's slave program output
slave_debug	2.3.4.85	Generate debugging output in the channel's slave program output
	Long address lists or headers	
expandchannel	2.3.4.16	Channel in which to perform deferred expansion due to application of expandlimit
expandlimit	2.3.4.16	Process an incoming message "off-line" when the number of addressees exceeds this limit
holdlimit	2.3.4.16	.HELD an incoming message when the number of addressees exceeds this limit
maxprocchars	2.3.4.79	Specify maximum length of headers to process
	Mailbox filters	
channelfilter	2.3.4.86	Specify the location of channel filter file for outgoing messages; synonym for destinationfilter
destinationfilter	2.3.4.86	Specify the location of channel filter file for outgoing messages
fileinto	2.3.4.86	Specify effect on address when a mailbox filter fileinto operation is applied
filter	2.3.4.86	Specify the location of user filter files
nochannelfilter	2.3.4.86	Do not perform channel filtering on outgoing messages; synonym for nodestinationfilter
nodestinationfilter	2.3.4.86	Do not perform channel filtering for outgoing messages
nofileinto	2.3.4.86	Mailbox filter fileinto operator has no effect
nofilter	2.3.4.86	Do not perform user mailbox filtering
nosourcefilter	2.3.4.86	Do not perform channel filtering for incoming messages
sourcefilter	2.3.4.86	Specify the location of channel filter file for incoming messages
	Notification messages and postmaster messages	
aliaspostmaster	2.3.4.63	Redirect postmaster messages to the local channel postmaster
copysendpost	2.3.4.21	Send copies of failures to the postmaster unless the originator address is blank

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		Notification messages and postmaster messages	
	copywarnpost	2.3.4.22	Send copies of warnings to the postmaster unless the originator address is blank
	errsendpost	2.3.4.21	Send copies of failures to the postmaster if the originator address is illegal
	errwarnpost	2.3.4.22	Send copies of warnings to the postmaster if the originator address is illegal
†	goldmail	2.3.4.26	Generate Gold-Mail compatible read receipts
	includefinal	2.3.4.27	Include final form of address in delivery notifications
†	**nogoldmail**	2.3.4.26	Do not generate Gold-Mail compatible read receipts
	nonurgentnotices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned for messages of non-urgent priority
	noreturnaddress	2.3.4.63	Use the RETURN_ADDRESS option value
	noreturnpersonal	2.3.4.63	Use the RETURN_PERSONAL option value
	normalnotices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned for messages of normal priority
	nosendpost	2.3.4.21	Do not send copies of failures to the postmaster
	notices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned
	nowarnpost	2.3.4.22	Do not send copies of warnings to the postmaster
	postheadbody	2.3.4.23	Both the message's header and body are sent to the postmaster when a delivery failure occurs
	postheadonly	2.3.4.23	Only the message's header is sent to the postmaster when a delivery failure occurs
	readreceiptmail	2.3.4.25	Ignore read receipt requests when delivering to VMS MAIL, rather than ''downgrading'' them to delivery receipt requests; leave it up to user agents to act upon the read receipt request
	reportboth	2.3.4.24	Generate both header and NOTARY delivery receipt requests from ''foreign'' delivery receipt requests
	reportheader	2.3.4.24	Generate only header delivery receipt requests from ''foreign'' delivery receipt requests
	reportnotary	2.3.4.24	Generate only NOTARY delivery receipt requests from ''foreign'' delivery receipt requests
	reportsuppress	2.3.4.24	Suppress delivery receipt requests from ''foreign'' delivery receipt requests
	returnaddress	2.3.4.63	Set the return address for the local Postmaster
	returnenvelope	2.3.4.64	Control use of blank envelope return addresses
	returnpersonal	2.3.4.63	Set the personal name for the local Postmaster
	sendpost	2.3.4.21	Send copies of failures to the postmaster
	suppressfinal	2.3.4.27	Include only original form of address in notification messages
	urgentnotices	2.3.4.20	Specify the amount of time which may elapse before notices are sent and messages returned for messages of urgent priority
	warnpost	2.3.4.22	Send copies of warnings to the postmaster

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
	Processing control and job submission		
	addrsperjob	2.3.4.14	Number of addresses to be processed by a single job
	after	2.3.4.18	Specify time delay before master channel programs run
	bidirectional	2.3.4.7	Channel is served by both a master and slave program
	deferred	2.3.4.19	Honor deferred delivery dates
	expandchannel	2.3.4.16	Channel in which to perform deferred expansion due to application of expandlimit
	expandlimit	2.3.4.16	Process an incoming message "off-line" when the number of addressees exceeds this limit
	filesperjob	2.3.4.14	Number of queue entries to be processed by a single job
	immediate	2.3.4.9	Delivery started immediately after submission for messages of second-class or higher priority
	immnonurgent	2.3.4.9	Delivery started immediately after submission even for messages with lower than normal priority
	immnormal	2.3.4.9	Delivery started immediately after submission for messages of normal or higher priority
	immurgent	2.3.4.9	Delivery started immediately after submission for urgent messages only
	master	2.3.4.7	Channel is served only by a master program
	maxjobs	2.3.4.14	Maximum number of jobs which can be created at once
	maxperiodicnonurgent	2.3.4.11	Specify that periodic jobs should only process messages of nonurgent or lower priority
	maxperiodicnormal	2.3.4.11	Specify that periodic jobs should only process messages of normal or lower priority
	maxperiodicurgent	2.3.4.11	Specify that periodic jobs should process messages of urgent or lower priority
	minperiodicnonurgent	2.3.4.11	Specify that periodic jobs should only process messages of nonurgent or higher priority
	minperiodicnormal	2.3.4.11	Specify that periodic jobs should only process messages of normal or higher priority
	minperiodicurgent	2.3.4.11	Specify that periodic jobs should only process messages of urgent priority
	nodeferred	2.3.4.19	Do not honor deferred delivery dates
	nonurgentblocklimit	2.3.4.10	Force messages above this size to wait unconditionally for a periodic job
†	nonurgentqueue	2.3.4.18	Specify the queue for master channel program processing of nonurgent messages
	normalblocklimit	2.3.4.10	Force messages above this size to nonurgent priority
†	normalqueue	2.3.4.18	Specify the queue for master channel program processing of normal messages
	noserviceall	2.3.4.12	Immediate delivery jobs process only the messages they were queued to process
	period	2.3.4.9	Specify periodicity of periodic channel service
	periodic	2.3.4.9	Channel is serviced only periodically; immediate delivery processing is never done
	queue	2.3.4.18	Specify queue master channel programs run in

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		Processing control and job submission	
	serviceall	2.3.4.12	Immediate delivery jobs process all messages (queued for the channel)
	slave	2.3.4.7	Channel is serviced only by a slave program
	threaddepth	2.3.4.29	Number of messages triggering new thread with multithreaded SMTP client
	urgentblocklimit	2.3.4.10	Force messages above this size to normal priority
†	urgentqueue	2.3.4.18	Specify the queue for master channel program processing of urgent messages
	user	<REFERENCE id="A22 CHANNEL_ORIGINATOR_ATTRIBUTE">	Specify account under which to run the pipe channel or specify message header mailbox name
		SASL and TLS	
	authrewrite	2.3.4.44	Use SMTP AUTH information in header
	client_auth	2.3.4.43	Specify which CLIENT_AUTH section to use for client SASL
	maysasl	2.3.4.43	Allow SMTP server and client SASL authentication
	maysaslclient	2.3.4.43	SMTP client attempts to use SASL authentication
	maysaslserver	2.3.4.43	SMTP server offers SASL authentication
§	maytls	2.3.4.45	SMTP client and server allow TLS use
§	maytlsclient	2.3.4.45	SMTP client will attempt TLS use
§	maytlsserver	2.3.4.45	SMTP server allows TLS use
	msexchange	2.3.4.46	Channel serves MS Exchange gateways
	mustsasl	2.3.4.43	Must use SASL authentication
	mustsaslclient	2.3.4.43	SMTP client insists upon SASL authentication
	mustsaslserver	2.3.4.43	SMTP server insists upon SASL authentication
§	musttls	2.3.4.45	SMTP client and server insist upon TLS use and will not transfer messages with remote sides that do not support TLS
§	musttlsclient	2.3.4.45	SMTP client insists upon TLS use and will not send messages to any remote SMTP server that does not support TLS use
§	musttlsserver	2.3.4.45	SMTP server insists upon TLS use and will not accept messages from any remote SMTP client that does not support TLS use
	nomsexchange	2.3.4.46	Channel does not serve MS Exchange gateways
	nosasl	2.3.4.43	SASL authentication not attempted or permitted
	nosaslclient	2.3.4.43	SMTP client does not attempt SASL authentication
	nosaslserver	2.3.4.43	SMTP server does not permit SASL authentication
	nosaslswitchchannel	2.3.4.43	Do not allow switching to this channel upon successful SASL authentication
§	**notls**	2.3.4.45	SMTP client and server neither attempt nor allow TLS use
§	notlsclient	2.3.4.45	SMTP client does not attempt TLS use when sending messages
§	notlsserver	2.3.4.45	SMTP server does not offer or allow TLS use when receiving messages
	saslswitchchannel	2.3.4.43	Switch to another channel when SASL authentication is successful

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		SASL and TLS	
§	**tlsswitchchannel**	2.3.4.45	Switch to specified channel upon successful TLS negotiation
		Sensitivity limits	
	sensitivitycompanyconfidential	2.3.4.88	Allow messages of any sensitivity
	sensitivitynormal	2.3.4.88	Reject messages whose sensitivity is higher than normal
	sensitivitypersonal	2.3.4.88	Reject messages whose sensitivity is higher than personal
	sensitivityprivate	2.3.4.88	Reject messages whose sensitivity is higher than private
		Size limits on messages, and user quotas and privileges	
	blocklimit	2.3.4.78	Maximum number of PMDF blocks allowed per message
	exquota	2.3.4.80	On OpenVMS, use EXQUOTA privileges if necessary to deliver VMS MAIL messages; on UNIX treat as holdexquota for Berkeley mailboxes; on all platforms deliver to overquota PMDF MessageStore or PMDF popstore accounts
	holdexquota	2.3.4.80	Hold messages for users that are over quota
	holdlimit	2.3.4.16	.HELD an incoming message when the number of addressees exceeds this limit
	linelimit	2.3.4.78	Maximum number of lines allowed per message
†	network	2.3.4.89	NETMBX privilege required for use
	noblocklimit	2.3.4.78	No limit specified for the number of PMDF blocks allowed per message
	noexquota	2.3.4.80	Return to originator any messages to users who are over quota
	nonurgentblocklimit	2.3.4.10	Force messages above this size to wait unconditionally for a periodic job
	normalblocklimit	2.3.4.10	Force messages above this size to nonurgent priority
	sourceblocklimit	2.3.4.78	Maximum number of PMDF blocks allowed per incoming message
	urgentblocklimit	2.3.4.78	Force messages above this size to normal priority

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

Keyword	Section	Usage
	SMTP commands and protocol	
allowetrn	2.3.4.34	Honor SMTP client ETRN commands
blocketrn	2.3.4.34	Do not honor SMTP client ETRN commands
checkehlo	2.3.4.32	Check the SMTP response banner for whether to use EHLO
disableetrn	2.3.4.34	Disable support for the ETRN SMTP command
domainetrn	2.3.4.34	Honor only those SMTP client ETRN commands that specify a domain
domainvrfy	2.3.4.35	Issue SMTP VRFY commands using full address
ehlo	2.3.4.32	Use EHLO on all initial SMTP connections
eightbit	2.3.4.50	Channel supports eight bit characters

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

Keyword	Section	Usage
		SMTP commands and protocol
eightnegotiate	2.3.4.50	Channel should negotiate use of eight bit transmission if possible
eightstrict	2.3.4.50	Channel should reject messages that contain unnegotiated eight bit data
localvrfy	2.3.4.35	Issue SMTP VRFY command using local address
loopcheck	2.3.4.91	Automatically detect mail loops when sending.
mailfromdnsverify	2.3.4.41	Verify that the domain specified on MAIL FROM: line is in the DNS
noehlo	2.3.4.32	Never use the SMTP EHLO command
nomailfromdnsverify	2.3.4.41	Do not perform DNS domain verification on the MAIL FROM: address
nosendetrn	2.3.4.33	Do not send SMTP ETRN command
nosmtp	2.3.4.31	Channel does not use SMTP
novrfy	2.3.4.35	Do not issue SMTP VRFY commands
sendetrn	2.3.4.33	Send SMTP ETRN command
sevenbit	2.3.4.50	Channel does not support eight bit characters; eight bit characters must be encoded
silentetrn	2.3.4.34	Honor SMTP client ETRN commands, without echoing channel information
smtp	2.3.4.31	Channel uses SMTP
smtp_cr	2.3.4.31	Accept CR as an SMTP line terminator
smtp_crlf	2.3.4.31	Require CRLF as the SMTP line terminator
smtp_crorlf	2.3.4.31	Allow any of CR, LF, or CRLF as the SMTP line terminator
smtp_lf	2.3.4.31	Accept LF as an SMTP line terminator
streaming	2.3.4.28	Specify degree of protocol streaming for channel to use
vrfyallow	2.3.4.36	Provide informative responses to SMTP VRFY command
vrfydefault	2.3.4.36	Default responses to SMTP VRFY command, according to channel's HIDE_VERIFY option setting
vrfyhide	2.3.4.36	Provide obfuscatory responses to SMTP VRFY command
		TCP/IP connections and DNS lookups
cacheeverything	2.3.4.13	Cache all connection information
cachefailures	2.3.4.13	Cache only connection failure information
cachesuccesses	2.3.4.13	Cache only connection success information
connectalias	2.3.4.5	Do not rewrite addresses upon message dequeue
connectcanonical	2.3.4.5	Rewrite addresses upon message dequeue
daemon	2.3.4.81	Specify name of a gateway daemon to route to
defaultmx	2.3.4.38	Channel determines whether or not to do MX lookups from network
defaultnameservers	2.3.4.38	Consult TCP/IP stack's choice of nameservers
forwardcheckdelete	2.3.4.40	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, delete the name and use the IP address

Table 2–6 (Cont.) Channel Block Keywords Grouped by Functionality

	Keyword	Section	Usage
		TCP/IP connections and DNS lookups	
	forwardchecknone	2.3.4.40	Do not perform a forward lookup after a DNS reverse lookup
	forwardchecktag	2.3.4.40	If a reverse DNS lookup has been performed, next perform a forward lookup on the returned name to check that the returned IP number matches the original; if not, tag the name with *
	identnone	2.3.4.40	Do not perform IDENT lookups; do perform IP to hostname translation; include both hostname and IP address in Received: header
	identnonelimited	2.3.4.40	Do not perform IDENT lookups; do perform IP to hostname translation, but do not use the hostname during channel switching; include both hostname and IP address in Received: header
	identnonenumeric	2.3.4.40	Do not perform IDENT lookups or IP to hostname translation
	identnonesymbolic	2.3.4.40	Do not perform IDENT lookups; do perform IP to hostname translation; include only the hostname in Received: header
	interfaceaddress	2.3.4.37	Bind to the specified TCP/IP interface address
	lastresort	2.3.4.39	Specify a last resort host
	mailfromdnsverify	2.3.4.41	Verify that the domain specified on MAIL FROM: line is in the DNS
	mx	2.3.4.38	TCP/IP network and software supports MX record lookups
	nameservers	2.3.4.38	Consult specified nameservers rather than TCP/IP stack's choice
	nocache	2.3.4.13	Do not cache any connection information
†	nodns	2.3.4.38	TCP/IP network does not support DNS (nameserver) lookups
	nomailfromdnsverify	2.3.4.41	Do not perform DNS domain verification on the MAIL FROM: address
	nomx	2.3.4.38	TCP/IP network does not support MX lookups
	nonrandommx	2.3.4.38	Do MX lookups; do not randomize returned entries with equal precedence
	port	2.3.4.37	Send to the specified TCP/IP port
	randommx	2.3.4.38	Do MX lookups; randomize returned entries with equal precedence
	threaddepth	2.3.4.29	Number of messages triggering new thread with multithreaded SMTP client
		Miscellaneous	
	description	2.3.4.87	Channel description
	submit	2.3.4.8	Mark the channel as a submit-only channel

2.3.4.1 Address Types and Conventions (822, 733, uucp, header_822, header_733, header_uucp)

This group of keywords controls what types of addresses the channel supports. A distinction is made between the addresses used in the transport layer (the message envelope) and those used in message headers.

822 (sourceroute)
Source route envelope addresses. This channel supports full RFC 822 format envelope addressing conventions including source routes. The keyword sourceroute is also available as a synonym for 822. This is the default if no other envelope address type keyword is specified.

733 (percents)
Percent sign envelope addresses. This channel supports full RFC 822 format envelope addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead. This keyword is also used in conjunction with DECnet MAIL-11 channels to force truncation of domain-style names after the first period. The keyword percents is also available as a synonym for 733.

Note: Use of 733 address conventions on an SMTP channel will result in these conventions being carried over to the transport layer addresses in the SMTP envelope. This may violate RFC 821. Only use 733 address conventions when you are sure they are necessary.

uucp (bangstyle)
Bang-style envelope addresses. This channel uses addresses that conform to RFC 976 bang-style address conventions in the envelope (*i.e.,* this is a UUCP channel). The keyword bangstyle is also available as a synonym for uucp.

header_822
Source route header addresses. This channel supports full RFC 822 format header addressing conventions including source routes. This is the default if no other header address type keyword is specified.

header_733
Percent sign header addresses. This channel supports RFC 822 format header addressing with the exception of source routes; source routes should be rewritten using percent sign conventions instead.

Percent sign routing is used for PhoneNet channels which connect to systems running older versions of PMDF.

Note: Use of 733 address conventions in message headers may violate RFC 822 and RFC 976. Only use this keyword if you are sure that the channel connects to a system that simply cannot deal with source route addresses.

header_uucp
UUCP or bang-style header addresses. The use of this keyword is *not* recommended. Such usage grossly violates RFC 976.

2.3.4.2 Address Interpretation (bangoverpercent, nobangoverpercent)

Addresses are always interpreted in accordance with RFC 822 and RFC 976. However, there are ambiguities in the treatment of certain composite addresses that are not addressed by these standards. In particular, an address of the form A!B%C can be interpreted as either A as the routing host and C as the final destination host, or C as the routing host and A as the final destination host.

While RFC 976 implies that it is all right for mailers to interpret addresses using the latter set of conventions, it does not say that such an interpretation is required. In fact, some situations may be better served by the former interpretation.

The bangoverpercent keyword forces the former A!(B%C) interpretation. The nobangoverpercent keyword forces the latter (A!B)%C interpretation. nobangoverpercent is the default. See Section 2.2.3.1 for further details.

Note: This keyword does not affect the treatment of addresses of the form A!B@C. These addresses are always treated as (A!B)@C. Such treatment is mandated by both RFC 822 and RFC 976.

2.3.4.3 Routing Information in Addresses (exproute, noexproute, improute, noimproute)

The ideal addressing model that PMDF deals with assumes that all systems are aware of the addresses of all other systems and how to get to them. Unfortunately, this ideal is not attainable in many cases. The usual exception occurs when a channel connects to one or more systems that are not known to the rest of the world (*e.g.,* internal machines on a private DECnet or TCP/IP network). Addresses for systems on this channel may not be legal on remote systems outside of the site. If such addresses are to be made repliable, they must contain a source route that tells remote systems to route messages through the local machine. The local machine can then (automatically) route the messages to these machines.

The exproute keyword (short for "explicit routing") tells PMDF that the associated channel requires explicit routing when its addresses are passed on to remote systems. If this keyword is specified on a channel, PMDF will add routing information containing the name of the local system (or the current alias for the local system) to all header addresses and all envelope From: addresses that match the channel. noexproute, the default, specifies that no routing information should be added.

The PMDF option EXPROUTE_FORWARD can be used to restrict the action of exproute to backward-pointing addresses if desired.

Another scenario occurs when PMDF connects to a system via a channel that cannot perform proper routing for itself. In this case all addresses associated with other channels need to have routing inserted into them when they are used in mail sent to the channel that connects to the incapable system.

Implicit routing and the improute keyword is used to handle this situation. PMDF knows that all addresses matching other channels need routing when they are used in mail sent to a channel marked improute. noimproute, the default, specifies that no

routing information should be added to addresses in messages going out on the specified channel.

The PMDF option IMPROUTE_FORWARD can be used to restrict the action of `improute` to backward-pointing addresses if desired.

The `exproute` and `improute` keywords should be used sparingly. It makes addresses longer, more complex, and may defeat intelligent routing schemes used by other systems.

Explicit and implicit routing should not be confused with specified routes. Specified routes are used to insert routing information from rewrite rules into addresses. This is activated by the special A@B@C rewrite rule template. Specified routes, when activated, apply to all addresses, both in the header and the envelope. Specified routes are activated by particular rewrite rules and as such are usually independent of the channel currently in use. Explicit and implicit routing, on the other hand, are controlled on a per-channel basis and the route address inserted is always the local system.

2.3.4.4 Short Circuiting Rewriting of Routing Addresses (`routelocal`)

The `routelocal` channel keyword causes PMDF, when rewriting an address to the channel, to attempt to "short circuit" any explicit routing in the address. Explicitly routed addresses (using !, %, or @ characters) will be simplified.

Use of this keyword on "internal" channels, such as internal TCP/IP channels, can allow simpler configuration of SMTP relay blocking.

Note that this keyword should not be used on channels that may require explicit % or other routing, such as PMDF-LAN or PMDF-MR channels.

2.3.4.5 Address Rewriting Upon Message Dequeue (`connectalias`, `connectcanonical`)

PMDF normally rewrites addresses as it enqueues messages to its channel queues. No additional rewriting is done during message dequeue. This presents a potential problem when host names change while there are messages in the channel queues still addressed to the old name.

The `connectalias` keyword tells PMDF to simply deliver to whatever host is listed in the recipient address. This is the default. `connectcanonical` forces PMDF to run the address through the rewrite rules one additional time and use the host that results.

`connectcanonical` should only be used specifically to deal with problems with queued messages—it may have unintended effects on other message traffic. In particular, `connectcanonical` must *not* be used on bit_ channels.

2.3.4.6 Channel-specific Rewrite Rules (rules, norules)

The rules keyword tells PMDF to enforce channel-specific rewrite rule checks for this channel. This is the default. norules tells PMDF not to check. These two keywords are usually used for debugging and are rarely used in actual applications.

2.3.4.7 Channel Directionality (master, slave, bidirectional)

Three keywords are used to specify whether a channel is served by a master program (master), a slave program (slave), or both (bidirectional). The default, if none of these keywords is specified, is bidirectional. These keywords determine whether PMDF bothers to initiate delivery activity when a message is queued to the channel — there is no point in doing this on a slave-only channel.

The use of these keywords reflects certain fundamental characteristics of the corresponding channel program or programs. The descriptions of the various channels PMDF supports indicate when and where these keywords should be used.

2.3.4.8 Channel Operation Type (submit)

PMDF supports RFC 2476's Message Submission protocol. The submit keyword may be used to mark a channel as a submit-only channel. This is normally useful mostly on TCP/IP channels, such as an SMTP server run on a special port used solely for submitting messages; RFC 2476 establishes port 587 for such message submission use.

2.3.4.9 Channel Service Periodicity (immediate, immnonurgent, immnormal, immurgent, periodic, period)

If a channel is capable of master-mode operations (as specified with the master keyword), such operations may be initiated either by a periodic service job or on demand as delivery is needed. The keyword periodic inhibits initiation of delivery jobs on demand for the channel it is associated with regardless of priority. The immediate keyword, which is the default, specifies that jobs should run on demand for messages of appropriate urgency; what appropriate urgency means is controlled via the keywords described below.

- immurgent enables immediate delivery processing on messages with a priority setting of urgent. Messages with a lower priority will wait for periodic processing.
- immnormal enables immediate delivery for messages with normal or urgent priority. immnormal is the default.
- immnonurgent enables immediate delivery for urgent, normal, and non-urgent messages.

Thus the default behavior (immediate immnormal) enables immediate processing for all but nonurgent or lower priority messages.

Delivery via periodic service jobs is always possible unless the channel is marked with the `slave` keyword. Channels capable of master-mode operation are periodically checked for pending messages by periodic service jobs. These jobs runs at fixed intervals — usually every four hours, though you may change this interval, if desired. On OpenVMS systems, the interval is changed by setting the system logical PMDF_POST_INTERVAL; if used, PMDF_POST_INTERVAL should be set to a string of the form *DD HH:MM:SS* (*e.g.,* "`0 00:30:00`"). On UNIX systems, the interval is determined in the `crontab` entry for the post job; see the appropriate edition of the *PMDF Installation Guide*.

Not all channels may need service at the same intervals. For example, a channel may see little traffic and be expensive to service (*i.e.,* it costs money to place a connecting phone call on a master-only periodic PhoneNet channel). Servicing such a channel at longer intervals than that of a single period between periodic jobs may lower the cost of operation without significantly affecting the quality of service. In another case, one particular channel may see very heavy traffic and may require frequent service, while other channels need servicing much less often. In this situation it may be appropriate to service the heavily used channel more often than any other.

The `period` keyword can be used to control how often a channel is serviced. This keyword must be followed by an integer value N. The channel is then serviced by every Nth service job. The default value of the `period` keyword is 1, which means that every periodic service job will check the channel for pending messages.

2.3.4.10 Message Size Affecting Priority (`urgentblocklimit`, `normalblocklimit`, `nonurgentblocklimit`)

The `urgentblocklimit`, `normalblocklimit`, and `nonurgentblocklimit` keywords may be used to instruct PMDF to downgrade the priority of messages based on size. The keywords must be followed by a single integer value, specifying the message size, in PMDF blocks, at which to perform the priority downgrading. A PMDF block is normally 1024 bytes; this can be changed with the BLOCK_SIZE option in the PMDF option file; see Section 7.3.5. This priority, in turn, may affect whether the message is processed immediately, or whether it is left to wait for processing until the next periodic job runs; see Section 2.3.4.9.

The `urgentblocklimit` keyword instructs PMDF to downgrade messages larger than the specified size to normal priority. The `normalblocklimit` keyword instructs PMDF to downgrade messages larger than the specified size to nonurgent priority. The `nonurgentblocklimit` keyword instructs PMDF to downgrade messages larger than the specified size to lower than nonurgent priority (second class priority), meaning that the messages will always wait for the next periodic job for further processing.

2.3.4.11 Priority of Messages to be Handled by Periodic Jobs (minperiodicnonurgent, minperiodicnormal, minperiodicurgent, maxperiodicnonurgent, maxperiodicnormal, maxperiodicurgent)

When periodic delivery jobs are used they normally process all messages queued for the channel. However, on some channels it may be desirable to limit normal periodic job processing to only messages of specified priorities. Other special site-supplied periodic jobs may then process the remaining messages. For instance, a site might choose to have normal PMDF periodic jobs pass over nonurgent messages, leaving those nonurgent messages to be delivered by some site-supplied job (perhaps scheduled to run at off-peak hours).

The minperiodicnonurgent, minperiodicnormal, or minperiodicurgent, keywords specify the minimum priority of message that a periodic job should try to deliver; the job will ignore messages of lower priority. The maxperiodicnonurgent, maxperiodicnormal, or maxperiodicurgent keywords specify the maximum priority of message that a periodic job should try to deliver; the job will ignore messages of higher priority.

2.3.4.12 Immediate Delivery Job Service Actions (serviceall, noserviceall)

When immediate delivery jobs are used they normally process only a subset of the messages queued for the channel. There may be other messages that were queued to the channel at some prior time that will not be processed. However, on some channels, particularly those that only provide a link to a single remote system, this sort of operation may be inappropriate: if the immediate delivery job is successful in connecting to the remote system it may be able to easily process all the messages that are queued.

The serviceall and noserviceall keywords control this behavior. noserviceall, the default, indicates that immediate delivery jobs should only process the messages they were queued to process. serviceall specifies that immediate jobs should attempt to process all messages queued to the channel.

It may be tempting to indulge in use of serviceall on most or all channels. Be warned, however, that use of serviceall is probably not suitable for most channels that connect to multiple remote systems, or channels that entail lots of per-message overhead. If serviceall is used on such channels it may cause a dramatic increase in network and message processing overhead and the net result may be slower message processing overall.

Note that these keywords do not change the order in which message processing occurs. Immediate jobs always attempt to process the messages they were created to process prior to turning to other messages that are also in the channel queue.

2.3.4.13 Channel Connection Information Caching (cacheeverything, cachesuccesses, cachefailures, nocache)

Channels using the SMTP protocol maintain a cache containing a history of prior connection attempts. This cache is used to avoid reconnecting multiple times to inaccessible hosts, which can waste lots of time and delay other messages.

The cache normally records both connection successes and failures. (Successful connection attempts are recorded in order to offset subsequent failures — a host that succeeded before but fails now doesn't warrant as long of a delay before making another connection attempt as does one that has never been tried or one that has failed previously.)

However, the caching strategy used by PMDF is not necessarily appropriate for all situations. For example, a channel that is used to connect to a single flakey host does not benefit from caching. Therefore channel keywords are provided to adjust PMDF's cache.

The `cacheeverything` keyword enables all forms of caching and is the default. `nocache` disables all caching. `cachefailures` enables caching of connection failures but not successes — this forces a somewhat more draconian retry than `cacheeverything` does. Finally, `cachesuccesses` caches only successes. This last keyword is effectively equivalent to `nocache` for SMTP channels.

2.3.4.14 Number of Addresses or Message Files to Handle per Service Job or File (addrsperjob, filesperjob, maxjobs)

PMDF normally creates one delivery service job per channel that needs service. This applies to both immediate service and periodic service jobs: when a message is initially sent and immediate service is needed one job is created for each channel to which the message is queued, and when PMDF creates periodic jobs it normally creates one periodic job for each channel that needs service.

A single service job may not be sufficient to insure prompt delivery of all messages, however. In particular, PMDF-FAX messages may take a long time to deliver; if multiple FAX modems are available it is not sensible to use a single job and a single modem.

The `addrsperjob` and `filesperjob` keywords can be used to cause PMDF to create additional service jobs. Each one of these keywords takes a single positive integer parameter which specifies how many addresses or queue entries (*i.e.,* files) must be sent to the associated channel before more than one service job is created to handle them. If a value less than or equal to zero is given it is interpreted as a request to queue only one service job. Not specifying a keyword is equivalent to specifying a value of zero. The effect of these keywords is maximized; the larger number computed will be the number of service jobs that are actually created.

The `addrsperjob` keyword computes the number of services jobs to start by dividing the total number of To: addressees in all entries by the given value. The `filesperjob` keyword divides the number of actual queue entries or files by the given value. Note that the number of queue entries resulting from a given message is controlled by a large number of factors, including but not limited to the use of the `single` and `single_sys` keywords and the specification of header-modifying actions in mailing lists.

The maxjobs keyword places an upper bound on the total number of service jobs that can be created. This keyword must be followed by an integer value; if the computed number of service jobs is greater than this value only maxjobs jobs will actually be created. The default for this value if maxjobs is not specified is 100. Normally maxjobs is set to a value that is less than or equal to the total number of jobs that can run simultaneously in whatever service queue or queues the channel uses.

For example, if a message with 4 recipient addresses is queued to a channel marked addrsperjob 2 and maxjobs 5 a total of 2 service jobs will be created. But if a message with 23 recipient addresses is queued to the same channel only 5 jobs will be created because of the maxjobs restriction.

Note that these keywords affect the creation of both periodic and immediate service jobs. In the case of periodic jobs the number of jobs created is calculated from the total number of messages in the channel queue. In the case of immediate service jobs the calculation is based only on the message being entered into the queue at the time.

2.3.4.15 Multiple Addresses (multiple, addrsperfile, single, single_sys)

PMDF allows multiple destination addresses to appear in each queued message. Some channel programs, however, may only be able to process messages with one recipient, or with a limited number of recipients, or with a single destination system per message copy. For example, the SMTP client programs for the TCP/IP channels only establish a connection to a single remote host in a given transaction, so only addresses to that host can be processed (this despite the fact that a single channel is typically used for all TCP/IP traffic). Another example is that some SMTP servers may impose a limit on the number of recipients they can handle at one time, and they may not handle errors in this area at all gracefully; similar concerns may also arise on the local channel when MAIL-11 connections are being used.

The keywords multiple, addrsperfile, single, and single_sys can be used to control how PMDF handles multiple addresses. single means that a separate copy of the message should be created for each destination address on the channel. single_sys creates a single copy of the message for each destination system used. multiple, the default, creates a single copy of the message for the entire channel. Note that at least one copy of each message is created for each channel the message is queued to, regardless of the keywords used.

The addrsperfile keyword is used to put a limit on the maximum number of recipients that can be associated with a single message file in a PMDF channel queue, thus limiting the number of recipients that will be processed in a single operation. This keyword requires a single integer argument specifying the maximum number of recipient addresses allowed in a message file; if this number is reached PMDF will automatically create additional message files to accomodate them. (The default multiple keyword corresponds to imposing no limit on the number of recipients in a message file.)

2.3.4.16 Expansion of Multiple Addresses (expandlimit, expandchannel, holdlimit)

Most PMDF channels support the specification of multiple recipient addresses in the transfer of each inbound message. The specification of many recipient addresses in a single message may result in delays in message transfer processing ("on-line" delays). If the delays are long enough, network timeouts can occur, which in turn can lead to repeated message submission attempts and other problems.

PMDF provides a special facility to force deferred ("off-line") processing if more than a given number of addresses are specified for a single message. Deferral of message processing can decrease on-line delays enormously. Note, however, that the processing overhead is deferred, not avoided completely.

This special facility is activated by using a combination of, for instance, the generic reprocessing channel and the `expandlimit` keyword. The `expandlimit` keyword takes an integer argument that specifies how many addresses should be accepted in messages coming from the channel before deferring processing. The default value is infinite if the `expandlimit` keyword is not specified. A value of 0 will force deferred processing on all incoming addresses from the channel.

The `expandlimit` keyword must *not* be specified on the local channel or the reprocessing channel itself; the results of such a specification are unpredictable.

The channel actually used to perform the deferred processing may be specified using the `expandchannel` keyword; the reprocessing channel is used by default, if `expandchannel` is not specified, but use of some other reprocessing or processing channel may be useful for special purposes. If a channel for deferred processing is specified via `expandchannel`, that channel should be a reprocessing or processing channel; specification of other sorts of channels may lead to unpredictable results.

The reprocessing channel, or whatever channel is used to perform the deferred processing, must be added to the PMDF configuration file in order for the `expandlimit` keyword to have any effect. If your configuration was built by the PMDF configuration utility, then you should already have a reprocessing channel. If not consult Section 26.7.

Extraordinarily large lists of recipient addresses are often a characteristic of so-called SPAM—junk e-mail. The `holdlimit` keyword tells PMDF that messages coming in the channel that result in more than the specified number of recipients should be marked as `.HELD` messages and enqueued to the reprocess channel (or to whatever channel is specified via the `expandchannel` keyword). As `.HELD` messages, the files will sit unprocessed in that PMDF queue area awaiting manual intervention by the PMDF postmaster.

2.3.4.17 Multiple Subdirectories (subdirs)

PMDF by default stores all messages queued to a channel as files in the directory PMDF_QUEUE:[*channel-name*] (OpenVMS), or /pmdf/queue/*channel-name*/ (UNIX). However, a channel which handles a large number of messages and tends to build up a large store of message files waiting for processing, *e.g.*, a TCP/IP channel, may get better performance out of the file system if those message files are spread across a number of subdirectories. The subdirs channel keyword provides this capability: it should be followed by an integer which specifies the number of subdirectories across which to spread messages for the channel, *e.g.*,

```
tcp_local single_sys smtp subdirs 10
```

The maximum value is 999.

2.3.4.18 Service Job Queue Usage and Job Deferral (queue, nonurgentqueue, normalqueue, urgentqueue, after)

PMDF creates service jobs to deliver messages. The queues where the jobs are created can be selected on a channel by channel basis by using the queue keyword. On OpenVMS, the queues used can also be selected by using the nonurgentqueue, normalqueue, and urgentqueue keywords. These *queue keywords must be followed by the name of the queue to which delivery jobs for the current channel should be queued. The name of the queue should not contain more than twelve characters.

On OpenVMS, different queue usage for messages of different priorities may be explicitly set using the nonurgentqueue, normalqueue, or urgentqueue keywords. Otherwise, the queue value (if any) will be used for all messages. If no *queue keyword is specified, then the queue used is the default queue, MAIL$BATCH on OpenVMS.

On Unix and Windows, different queue usage for different channels is normally set up using the Job Controller configuration file. The queue channel keyword may be used on the channel to specify a queue that is defined in the Job Controller. However, if a queue is specified in the Job Controller configuration file, that one takes precedence.

VMS

Using multiple queues is especially useful when PMDF is run in a cluster. Certain channels may require hardware or software that is only available on a specific system within the cluster. Accordingly, queues may be associated with specific systems, making it possible to ensure that the jobs servicing a particular channel only run on the proper machine.

Execution of service jobs can be deferred using the after keyword. The after keyword must be followed by a specification of the amount of time to delay. If the value following the keyword is an unsigned integer value, it is interpreted as a number of seconds by which to defer the execution of the job—a delta time value.

VMS

On OpenVMS, there is an alternative time format specification. Anything other than an unsigned integer will be interpreted as being in standard OpenVMS combination date/time format. The specification must not be quoted and may not contain any spaces. For example, TOMORROW+1 delays execution of any submitted job to 1:00AM the following day; +00:01:00 delays execution of any submitted job for one minute. The specification

-:: is equivalent to the current date/time and is the default, resulting in immediate eligibility for execution.

VMS

On OpenVMS, deferred execution with an absolute time value is most often used to schedule delivery at a time when some resource is known to be available. It can also be used to defer delivery to a time when the system isn't so heavily loaded, but other techniques, such as placing dynamic limits on the allowed number of simultaneous jobs, may be a better solution.

Deferred execution with a (typically small) delta time value is most often used to increase throughput (*e.g.*, as a result of cutting down on image activation overhead) for heavily used channels. Note that, regardless of the `after` channel keyword, PMDF will not submit a new channel job when there is already a pending or holding job for that channel. (PMDF will, however, submit a new channel job when there are already running jobs for that channel.) So by using the `after` channel keyword to introduce a slight latency in the execution of immediate PMDF channel jobs, each such job has a window of time during which to "collect" all the messages sent to the channel in that time. Whereas normally an immediate PMDF channel job might typically handle only one (or at especially busy times perhaps two or three) messages, such use of the `after` channel keyword allows immediate PMDF channel jobs to typically collect and handle larger numbers of messages. For channels with high connection or image activation overhead, this can result in significantly higher overall throughput.

2.3.4.19 Deferred Delivery Dates (`deferred`, `nodeferred`)

The `deferred` channel keyword implements recognition and honoring of the Deferred-delivery: header. Messages with a deferred delivery date in the future will be held in the channel queue until they either expire and are returned or the deferred delivery date is reached. See RFC 2156 for details on the format and operation of the Deferred-delivery: header.

`nodeferred` is the default. It is important to realize that while support for deferred message processing is mandated by RFC 2156, actual implemention of it effectively lets people use the mail system as an extension of their disk quota.

2.3.4.20 Undeliverable Message Notification Times (`notices`, `nonurgentnotices`, `normalnotices`, `urgentnotices`)

The `notices`, `nonurgentnotices`, `normalnotices`, and `urgentnotices` keywords control the amount of time an undeliverable message is silently retained in a given channel queue. PMDF is capable of returing a series of warning messages to the originator and, if the message remains undeliverable, PMDF will eventually return the entire message.

Different return handling for messages of different priorities may be explicitly set using the `nonurgentnotices`, `normalnotices`, or `urgentnotices` keywords. Otherwise, the `notices` keyword values will be used for all messages.

The keyword is followed by a list of up to five monotonically increasing integer values. These values refer to the message ages at which warning messages are sent. The ages have units of days if the PMDF option RETURN_UNITS is 0 or not specified in the PMDF option file, or hours if the PMDF option RETURN_UNITS is 1. When an undeliverable message attains or exceeds the last listed age, it is returned (*i.e.,* bounced). When it attains any of the other ages, a warning notice is sent. The default if no `notices` keyword is given is to use the `notices` setting for the local, l, channel. If no setting has been made for the local channel, then the defaults 3, 6, 9, 12 are used meaning that warning messages are sent when the message attains the ages 3, 6, and 9 days (or hours) and the message is returned after remaining in the channel queue for more than 12 days (or hours).

The syntax for the `notices` keyword uses no punctuation. For example, the default return policy would be expressed as follows:

```
notices 3 6 9 12
```

If you want to change the notification ages for all of your channels, then the simplest thing to do is to add a defaults channel block to the start of the channel block section of your PMDF configuration file or to add the `notices` setting to your local channel. For instance,

```
defaults notices 1 3 6 9 12

l defragment charset7 us-ascii charset8 dec-mcs
example.com
```

The defaults channel would appear immediately after the first blank line in the PMDF configuration file, usually `PMDF_TABLE:pmdf.cnf` (OpenVMS) or `/pmdf/table/pmdf.cnf` (UNIX) or `C:\pmdf\table\pmdf.cnf` (NT). It is important that a blank line appear *before and after* the line "`defaults notices...`". See Section 2.3.5 for a full description of the defaults channel.

See Section 1.4.4 for more information on the `*notices` keywords and how they interact with return message processing.

2.3.4.21 Returned Messages (`sendpost`, `nosendpost`, `copysendpost`, `errsendpost`)

A channel program may be unable to deliver a message due to long-term service failures or invalid addresses. When this happens the PMDF channel program returns the message to the sender with an accompanying explanation of why the message was not delivered. PMDF will also optionally send a copy of all failed messages to the local postmaster. This is useful for monitoring message failures, but it can result in lots of traffic for the postmaster to deal with.

The keywords `sendpost`, `copysendpost`, `errsendpost`, and `nosendpost` are used to control the sending of failed messages to the postmaster. `sendpost` tells PMDF to send a copy of all failed messages to the postmaster unconditionally. `copysendpost` instructs PMDF to send a copy of the failure notice to the postmaster unless the originator address on the failing message is blank; *i.e.*, the postmaster gets copies of all failed messages except those messages that are actually themselves bounces or notifications. `errsendpost` instructs PMDF to only send a copy of the failure notice to the postmaster

when the notice cannot be returned to the originator. No failed messages are ever sent to the postmaster if `nosendpost` is specified. The default if none of these keywords is specified is to send a copy of failed mail messages to the postmaster unless error returns are completely suppressed with a blank Errors-to: header or a blank envelope From: address. Note that this default behavior does not correspond to any of the keyword settings.

2.3.4.22 Warning Messages (`warnpost`, `nowarnpost`, `copywarnpost`, `errwarnpost`)

In addition to returning messages, PMDF sometimes sends warnings detailing messages that it has been unable to deliver. This is generally due to timeouts based on the setting of the `notices` channel keyword, although in some cases channel programs may produce warning messages after failed delivery attempts. The warning messages contain a description of what's wrong and how long delivery attempts will continue. In most cases they also contain the headers and the first few lines of the message in question.

PMDF will also optionally send a copy of all warning messages to the local postmaster. This can be somewhat useful for monitoring the state of the various PMDF queues, although it does result in lots of traffic for the postmaster to deal with.

The keywords `warnpost`, `copywarnpost`, `errwarnpost`, and `nowarnpost` are used to control the sending of warning messages to the postmaster. `warnpost` tells PMDF to send a copy of all warning messages to the postmaster unconditionally. `copywarnpost` instructs PMDF to send a copy of the warning to the postmaster unless the originator address on the as yet undelivered message is blank; *i.e.*, the postmaster gets copies of all warnings of undelivered messages except for those as yet undelivered messages that are actually themselves bounces or notifications. `errwarnpost` instructs PMDF to only send a copy of the warning to the postmaster when the notice cannot be returned to the originator. No warning messages are ever sent to the postmaster if `nowarnpost` is specified. The default if none of these keywords is specified is to send a copy of warnings to the postmaster unless warnings are completely suppressed with a blank Warnings-to: header or a blank envelope From: address. Note that this default behavior does not correspond to any of the keyword settings.

2.3.4.23 Postmaster Returned Message Content (`postheadonly`, `postheadbody`)

When a PMDF channel program or the periodic message return job returns messages to both the postmaster and the original sender, the postmaster copy can either be the entire message or just the headers. Restricting the postmaster copy to just the headers adds an additional level of privacy to user mail. Note, however, this by itself does not guarantee message security; postmasters and system managers are typically in a position where the contents of messages can be read using system privileges if they so choose.

The keywords `postheadonly` and `postheadbody` are used to control what gets sent to the postmaster. `postheadbody` returns both the headers and the contents of the message. It is the default. `postheadonly` causes only the headers to be sent to the postmaster.

2.3.4.24 Delivery Receipt Request Style (reportboth, reportheader, reportnotary, reportsuppress)

The reportboth, reportheader, reportnotary, and reportsuppress channel keywords control which sort, if any, of delivery receipt request PMDF constructs from "foreign" delivery receipt requests, such as for messages coming in to PMDF via PMDF-LAN, PMDF-MR, PMDF-X400, or PMDF-MB400 channels, or via the addressing channel. On OpenVMS, these keywords also control the interpretation of delivery receipt requests from VMS MAIL or PMDF MAIL via L or D channels. For PMDF-LAN, PMDF-MR, PMDF-X400, and PMDF-MB400 channels, these keywords also control which sort of delivery receipt request PMDF will convert into the respective "foreign" delivery receipt request. (In the case of PMDF-X400, note that the keyword on the MIME_TO_X400 channel controls the behavior in both directions, to and from the X.400 world.) The current default is reportheader meaning to turn "foreign" delivery receipt requests into the old ad-hoc header style delivery receipt requests. reportnotary requests that only NOTARY[f] style delivery receipt requests be generated; this may become the default in a future version of PMDF. reportboth causes PMDF to generate both a header style and a NOTARY style delivery receipt request when seeing a "foreign" delivery receipt request; setting this may result in two delivery receipts from MTAs that support both forms of delivery receipt request. reportsuppress causes PMDF to ignore (suppress) incoming "foreign" delivery receipt requests.

2.3.4.25 Passing Read Receipt Requests to the VMS MAIL Mailbox (OpenVMS) (readreceiptmail)

Since the VMS MAIL mailbox does not support read receipts, by default PMDF "downgrades" read receipt requests into delivery receipt requests when delivering to the VMS MAIL mailbox. However, some clients—such as IMAP clients when the VMS MAIL mailbox is served out by an IMAP server—may have some read receipt support themselves, handled outside of the VMS MAIL message store itself; when a site has such clients in use it may be desired to "pass through" any read receipt requests rather than "downgrading" them to delivery receipt requests.

The readreceiptmail channel keyword when placed on the l (lowercase "L") channel causes PMDF to pass through read receipt requests.

2.3.4.26 Gold-Mail Compatible Read Receipts (OpenVMS) (goldmail, nogoldmail)

On OpenVMS systems, the Gold-Mail mail user agent, a product of Data Processing Design, Inc., is a commonly used replacement for VMS MAIL.

Gold-Mail provides some added functionality that is not available in regular VMS MAIL. In particular, Gold-Mail supports read receipts. PMDF is capable of converting its own read receipt requests into a format that is compatible with Gold-Mail. In order to activate this feature, the goldmail channel keyword should be used on the channel that delivers mail to Gold-Mail. This is usually the local channel, although it is possible for Gold-Mail delivery to be done on d or mail_ channels in some configurations.

[f] See RFC 1891.

The nogoldmail keyword disables this feature. nogoldmail is the default.

2.3.4.27 Including Altered Addresses in Notification Messages (includefinal, suppressfinal)

When PMDF generates a notification message (bounce message, delivery receipt message, *etc.*), there may be both an "original" form of a recipient address and an altered "final" form of that recipient address available to PMDF. PMDF always includes the original form (assuming it is present) in the notification message, since that is the form that the recipient of the notification message (the sender of the original message which the notification message concerns) is most likely to recognize. The includefinal and suppressfinal channel keywords control whether PMDF also includes the final form of the address. Suppressing the inclusion of the final form of address may be of interest to sites that are "hiding" their internal mailbox names from external view; such sites may prefer that only the original, "external" form of address be included in notification messages. includefinal is the default and means to include the final form of the recipient address; suppressfinal causes PMDF to suppress the final address form, if an original address form is present, from notification messages.

2.3.4.28 Protocol Streaming (streaming)

Some mail protocols support streaming operations. This means that PMDF can issue more than one operation at a time and wait for replies to each operation to arrive in batches. The streaming keyword controls the degree of protocol streaming used in the protocol associated with a channel. This keyword requires an integer parameter; how the parameter is interpreted is specific to the protocol in use.

Currently PMDF only supports the *experimental* use of streaming on SMTP channels. Implementation of this feature is experimental; it may change in future PMDF releases.

The streaming values available range from 0 to 3. A value of 0 specifies no streaming, a value of 1 causes groups of RCPT TO commands to stream, a value of 2 causes MAIL FROM/RCPT TO to stream, and a value of 3 causes HELO/MAIL FROM/RCPT TO or RSET/MAIL FROM/RCPT TO streaming to be used. The default value is 0.

Some SMTP implementations are known to react badly to streaming. In particular, sendmail is known to be incapable of handling streaming levels greater than 1. PMDF's server implementation of SMTP should work properly at any streaming level.

2.3.4.29 Triggering New Threads in Multi-threaded SMTP Channel (threaddepth)

The multithreaded SMTP client sorts outgoing messages to different destinations to different threads. The threaddepth keyword may be used to instruct PMDF's multithreaded SMTP client to handle only the specified number of messages in any one thread, using additional threads even for messages all to the same destination (hence normally all handled in one thread).

Use of threaddepth may be of particular interest for achieving multithreading on a daemon router TCP/IP channel—a TCP/IP channel that connects to a single specific SMTP server – when the SMTP server to which the channel connects can handle multiple simultaneous connections.

2.3.4.30 PMDF Channel Queue Directories' Locations (logicaldisk, nologicaldisk)

Note: The logicaldisk keyword is only supported on OpenVMS. However, symbolic links can be used on UNIX to provide similar functionality.

The logicaldisk channel keyword is used to spread PMDF channel queues across multiple disks. Specifically, when logicaldisk is specified on a channel, the name of the corresponding channel queue changes from PMDF_QUEUE:[*channel-name...*] to PMDF_QUEUE_*channel-name*:[*...]. nologicaldisk is the default.

Use of logicaldisk therefore requires that a corresponding PMDF_QUEUE_ *channel-name* logical be defined. Note that the logical name must be defined before pmdf cnbuild is run, because pmdf cnbuild checks for both the logicaldisk channel keyword and the existence of the logical name before it will enable this functionality.

2.3.4.31 Channel Protocol Selection (smtp, nosmtp)

These options specify whether or not a channel supports the SMTP protocol and what type of SMTP line terminator PMDF expects to see as part of that protocol. nosmtp means that the channel doesn't support SMTP; all the rest of these keywords imply SMTP support.

The selection of whether or not to use the SMTP protocol is implicit for most channels; the correct protocol is chosen by the use of the appropriate channel program or programs.

The keyword smtp is mandatory for all SMTP channels.

The keywords smtp_cr, smtp_crlf, smtp_crorlf, and smtp_lf may be used on SMTP channels to specify what character sequences to accept as line terminators. smtp_crlf means that lines must be terminated with a carriage return (CR) line feed (LF) sequence. smtp_crorlf or smtp means that lines may be terminated with any of a carriage return (CR), or a line feed (LF) sequence, or a full CRLF. smtp_lf means that a LF without a preceeding CR will be accepted. Finally, smtp_cr means that a CR will be accepted without a following LF. It is normal to use CRLF sequences as the SMTP line terminator, and this is what PMDF itself always generates; this option only affects PMDF's handling of incoming material.

2.3.4.32 SMTP EHLO Command (ehlo, checkehlo, noehlo)

The SMTP protocol has been extended (RFC 1869) to allow for negotiation of additional commands. This is done via the new EHLO command, which replaces RFC 821's HELO command. Extended SMTP servers respond to EHLO by providing a list of the extensions they support. Unextended servers return an unknown command error and the client then sends the old HELO command instead.

This fallback strategy normally works well with both extended and unextended servers. Problems can arise, however, with servers that do not implement SMTP according to RFC 821. In particular, some incompliant servers are known to drop the connection on receipt of an unknown command.

The PMDF client implements a strategy whereby it will attempt to reconnect and use HELO when any server drops the connection on receipt of an EHLO. However, this strategy still may not work if the remote server not only drops the connection but also goes into a problematic state upon receipt of EHLO.

The channel keywords ehlo, noehlo, and checkehlo are provided to deal with such situations. ehlo tells PMDF to use the EHLO command on all initial connection attempts. noehlo disables all use of the EHLO command. checkehlo tests the response banner returned by the remote SMTP server for the string "ESMTP". If this string is found EHLO is used; if not HELO is used. The default behavior is to use EHLO on all initial connection attempts, unless the banner line contains the string "fire away", in which case HELO is used; note that there is no keyword corresponding to this default behavior, which lies between the behaviors resulting from the ehlo and checkehlo keywords.

2.3.4.33 Sending an SMTP ETRN Command (sendetrn, nosendetrn)

The extended SMTP command ETRN (RFC 1985) allows an SMTP client to request that a remote SMTP server start up processing of the remote side's message queues destined for sending to the original SMTP client; that is, it allows an SMTP client and SMTP server to negotiate "switching roles", where the side originally the sender becomes the receiver, and the side originally the receiver becomes the sender. Or in other words, ETRN provides a way to implement "polling" of remote SMTP systems for messages incoming to one's own system. This can be useful for systems that only have transient connections between each other, for instance, over dial up lines. When the connection is brought up and one side sends to the other, via the ETRN command the SMTP client can also tell the remote side that it should now try to deliver any messages that need to travel in the reverse direction.

The SMTP client specifies on the SMTP ETRN command line the name of the system to which to send messages (generally the SMTP client system's own name). If the remote SMTP server supports the ETRN command, it will trigger execution of a separate process to connect back to the named system and send any messages awaiting delivery for that named system.

The sendetrn and nosendetrn channel keywords control whether the PMDF SMTP client sends an ETRN command at the beginning of an SMTP connection. The default is nosendetrn, meaning that PMDF will not send an ETRN command. The sendetrn keyword tells PMDF to send an ETRN command, if the remote SMTP server says it

supports ETRN. The sendetrn keyword should be followed by the name of the system requesting that its messages receive a delivery attempt.

2.3.4.34 Receiving an SMTP ETRN Command (allowetrn, blocketrn, disableetrn, domainetrn, silentetrn)

The allowetrn, blocketrn, disableetrn, domainetrn, and silentetrn keywords control PMDF's response when a sending SMTP client issues the SMTP ETRN command, requesting that PMDF attempt to deliver messages in the PMDF queues. allowetrn means that PMDF will attempt to honor all ETRN commands and will echo back the name of the channel that will be run in response to the ETRN command. silentetrn tells PMDF to honor all ETRN commands, but without echoing back the name of the channel which the domain matched and which PMDF will hence be attempting to run. domainetrn tells PMDF to honor only those ETRN commands that specify a domain with an @; it also causes PMDF not to echo back the name of the channel which the domain matched and which PMDF will hence be attempting to run. disableetrn disables support for the ETRN command entirely; ETRN will not be advertised by the SMTP server as a supported command. The default behavior, if none of these keywords is explicitly specified, corresponds most closely to silentetrn.

blocketrn tells PMDF not to honor an ETRN command if the ETRN command attempts to run that channel. Note that this keyword is therefore relevant on a destination channel, not on the incoming TCP/IP channel (unless that incoming channel would also be the destination channel for an attempted ETRN command).

See also Section 21.1.2.2 for a discussion of the ALLOW_ETRNS_PER_SESSION channel option, which may be used to limit the number of ETRN commands which PMDF will honor during a single session.

2.3.4.35 Sending an SMTP VRFY Command (domainvrfy, localvrfy, novrfy)

These keywords control PMDF's use of the VRFY command in its SMTP client. Under normal circumstances there is no reason for PMDF to issue a VRFY command as part of an SMTP dialogue — the SMTP MAIL TO command should perform the same function that VRFY does and return an appropriate error. However, SMTP servers exist that will accept any address in a MAIL TO (and bounce it later), whereas they perform more extensive checking as part of a VRFY command.

Therefore PMDF can be configured to issue SMTP VRFY commands. The keyword domainvrfy causes PMDF to issue a VRFY command with a full address (*e.g.*, *user@host*) as its argument. The localvrfy keyword causes PMDF to issue a VRFY command with just the local part of the address (*e.g.*, *user*). novrfy is the default.

2.3.4.36 Responding to SMTP VRFY Commands (vrfyallow, vrfydefault, vrfyhide)

These keywords control the PMDF SMTP server's response when a sending SMTP client issues an SMTP VRFY command. The vrfyallow keyword tells PMDF to issue a detailed, informative response. The vrfydefault tells PMDF to provide a detailed, informative response, unless the channel option HIDE_VERIFY=1 has been specified, as discussed in Section 21.1.2.2. The vrfyhide keyword tells PMDF to issue only a vague, ambiguous response. Thus these keywords allow per-channel control of VRFY responses, as opposed to the HIDE_VERIFY option which normally applies to all incoming TCP/IP channels handled via the same SMTP server.

2.3.4.37 TCP/IP Port Number and Interface Address (interfaceaddress, port)

SMTP over TCP/IP channels normally connect to the remote system's port 25 when sending messages. The port keyword may be used to instruct an SMTP over TCP/IP channel to connect to a non-standard port.

The interfaceaddress keyword controls the address to which a TCP/IP channel binds as the source address for outbound connections; that is, on a system with multiple interface addresses this keyword controls which address will be used as the source IP address when PMDF sends outgoing SMTP messages. Note that it complements the Dispatcher option INTERFACE_ADDRESS, Section 11.3.2, which controls which interface address a TCP/IP channel listens on for accepting incoming connections and messages.

2.3.4.38 TCP/IP Nameserver and MX Record Support (mx, nomx, nodns, defaultmx, randommx, nonrandommx, nameservers, defaultnameservers)

Some TCP/IP networks support the use of MX (mail forwarding) records and some do not. PMDF TCP/IP channel programs can be configured to not use MX records if they are not provided by the network to which the PMDF system is connected. Some PMDF TCP/IP channel programs can be configured to not do DNS (nameserver) lookups at all. randommx specifies that MX lookups should be done and MX record values of equal precedence should be processed in random order. nonrandommx specifies that MX lookups should be done and MX values of equal precedence should be processed in the same order in which they were received. The mx keyword is currently equivalent to nonrandommx; it may change to be equivalent to randommx in a future PMDF release. The nomx keyword disables MX lookups. The defaultmx keyword specifies that mx should be used if the network says that that MX records are supported.

defaultmx is the default on channels that support MX lookups in any form.

On UNIX, whether the underlying TCP/IP package's local host tables are used in addition to the DNS for lookups is up to the underlying TCP/IP package configuration. Generally, TCP/IP packages are configured so that local host tables will indeed be consulted. Consult your TCP/IP package documentation for details.

VMS

On OpenVMS, for PMDF's multithreaded TCP/IP channels, the underlying TCP/IP package's local host tables are not normally consulted during name lookups. In particular, note that specifying `nomx` on such a channel does not cause name lookups to refer to the underlying TCP/IP package's local host tables; specifying `nomx` merely causes such a channel to limit its DNS queries by not using MX records. But there is another keyword `nodns`, which goes further and disables DNS (nameserver) lookups entirely, causing all lookups to be done using the underlying TCP/IP package's local host tables; this keyword is only applicable on OpenVMS and only to multithreaded TCP/IP channels.

When nameserver lookups are being performed, that is, unless the `nodns` channel keyword is used on OpenVMS, or the `nsswitch.conf` file on UNIX or the NT TCP/IP configuration selects no use of nameservers, then the `nameserver` channel keyword may be used to specify a list of nameservers to consult rather than consulting the TCP/IP stack's own choice of nameservers. `nameservers` requires a space separated list of IP addresses for the nameservers, *e.g.*,

```
nameservers 1.2.3.1 1.2.3.2
```

`defaultnameservers` is the default, and means to use the TCP/IP stack's own choice of nameservers.

2.3.4.39 Specify a Last Resort Host (`lastresort`)

The `lastresort` keyword is used to specify a host to which to connect when all other connection attempts fail. In effect this acts as an MX record of last resort. This is only useful on SMTP channels.

The keyword requires a single parameter specifying the name of the "system of last resort", *e.g.*,

```
tcp_local single_sys smtp mx lastresort mailhub.example.com
TCP-DAEMON
```

2.3.4.40 Reverse DNS lookups on incoming SMTP connections (`identnone`, `identnonelimited`, `identnonenumeric`, `identnonesymbolic`, `forwardchecknone`, `forwardchecktag`, `forwardcheckdelete`)

The `identnone` keyword enables IP to hostname translation, and both IP number and hostname will be included in the Received: header for the message. The `identnonesymbolic` keyword enables IP to hostname translation; only the hostname will be included in the Received: header for the message. The `identnonenumeric` keyword inhibits the usual DNS reverse lookup translation of IP number to hostname, and may therefore result in a performance improvement at the cost of less user-friendly information in the Received: headers. `identnone` is the default.

The `identnonelimited` keyword has the same effect as `identnone`, as far as reverse DNS lookups, and information displayed in Received: header lines. Where they differ is that with `identnonelimited` the IP literal address is always used as the sole basis for any channel switching due to use of the `switchchannel` keyword, regardless of

whether the DNS reverse lookup succeeds in determining a host name. Note that since channel switching is always performed preferentially based on IP address rather than host name, the effect of `identnonelimited` is merely to disable ever trying host name switching in case all IP address rewriting failed.

Keyword	DNS reverse lookup	IP address in Received: header line	Reverse hostname in Received: header line	Fall back to hostname channel switch
`identnone`	Yes	Yes	Yes	Yes
`identnonelimited`	Yes	Yes	Yes	No
`identnonenumeric`	No	Yes	No	No
`identnonesymbolic`	Yes	No	Yes	Yes

The `forwardchecknone`, `forwardchecktag`, and `forwardcheckdelete` channel keywords can modify the effects of doing reverse lookups, controlling whether PMDF does a forward lookup of an IP name found via a DNS reverse lookup, and if such forward lookups are requested what PMDF does in case the forward lookup of the IP name does not match the original IP number of the connection. The `forwardchecknone` keyword is the default, and means that no forward lookup is done. The `forwardchecktag` keyword tells PMDF to do a forward lookup after each reverse lookup and to tag the IP name with an asterisk, `*`, if the number found via the forward lookup does not match that of the original connection. The `forwardcheckdelete` keyword tells PMDF to do a forward lookup after each reverse lookup and to ignore (delete) the reverse lookup returned name if the forward lookup of that name does not match the original connection IP address, and stick with the original IP address instead. (Note that having the forward lookup not match the original IP address is normal at many sites, where a more "generic" IP name is used for several different IP addresses.)

These keywords are only useful on SMTP channels that run over TCP/IP.

2.3.4.41 Verify that the domain on the MAIL FROM: line is in the DNS (`mailfromdnsverify`, `nomailfromdnsverify`)

Setting `mailfromdnsverify` on an incoming TCP/IP channel causes PMDF to verify that an entry in the DNS exists for the domain used on the SMTP MAIL FROM: command, and to reject the message if no such entry exists. `nomailfromdnsverify` is the default, and means that no such check is performed.

Note that performing DNS checks on the return address domain may result in rejecting some desired valid messages (for instance, from legitimate sites that simply have not yet registered their domain name, or at times of bad information in the DNS); it is contrary to the spirit of being generous in what you accept and getting the e-mail through, expressed in RFC 1123, Requirements for Internet Hosts. However, some sites may desire to perform such checks in cases where junk e-mail (SPAM) is being sent with forged e-mail addresses from non-existant domains.

2.3.4.42 Select an alternate channel for incoming mail (switchchannel, allowswitchchannel, noswitchchannel)

When a PMDF server accepts an incoming connection from a remote system it must choose a channel with which to associate the connection. Normally this decision is based on the transport used; for example, an incoming TCP/IP connection is automatically associated with the tcp_local channel.

This convention breaks down, however, when multiple outgoing channels with different characteristics are used to handle different systems over the same transport. When this is done incoming connections are not associated with the same channel as outgoing connections, and the result is that the corresponding channel characteristics are not associated with the remote system.

The switchchannel keyword provides a way to eliminate this difficulty. If switchchannel is specified on the initial channel the server uses, the IP address of the connecting (originating) host will be matched against the channel table and if it matches the source channel will change accordingly. If no IP address match is found or if a match is found that matches the original default incoming channel, PMDF may optionally try matching using the host name found by doing a DNS reverse lookup; see Section 2.3.4.40. The source channel may change to any channel marked switchchannel or allowswitchchannel (the default). noswitchchannel specifies that no channel switching should be done to or from the channel.

Specification of switchchannel on anything other than a channel that a server associates with by default will have no effect. At present switchchannel only affects SMTP channels, but there are actually no other channels where switchchannel would be reasonable. In particular, PhoneNet channels never need to switch since they are inherently point-to-point.

2.3.4.43 SMTP authentication and SASL (client_auth, maysasl, maysaslclient, maysaslserver, mustsasl, mustsaslclient, mustsaslserver, nosasl, nosaslclient, nosaslserver, saslswitchchannel, nosaslswitchchannel)

The client_auth, maysasl, maysaslclient, maysaslserver, mustsasl, mustsaslclient, mustsaslserver, nosasl, nosaslclient, nosaslserver, saslswitchchannel, and nosaslswitchchannel channel keywords are used to configure SASL use, specifically the use of the AUTH command, during the SMTP protocol by SMTP based channels such as TCP/IP channels. nosasl is the default, and means that SASL authentication will not be permitted (by the server) or attempted (by the client). It subsumes both nosaslserver and nosaslclient. Specifying maysaslserver causes the SMTP server to permit clients to attempt to use SASL authentication. Specifying maysaslclient causes the SMTP server to attempt to use SASL authentication. maysasl subsumes both maysaslserver and maysaslclient. Specifying mustsaslserver causes the SMTP server to insist that clients use SASL authentication; the SMTP server will not accept messages unless the remote client successfully authenticates. Specifying mustsaslclient causes the SMTP client to use SASL authentication, and it will refuse to continue with the SMTP connection if it cannot successfully authenticate. mustsasl subsumes both mustsaslserver and mustsaslclient. The saslswitchchannel keyword is used to cause incoming connections to be switched to a specified channel upon a client's successful SASL use. It takes a required value,

specifying the channel to which to switch. nosaslswitchchannel is the default, and means that channel switching is not performed upon a client's successful SASL use. The client_auth keyword is used for client-side SASL authentication to tell the TCP/IP channel which CLIENT_AUTH section of the security.cnf file to read to get the username and password to use to authenticate to the remote system. If this option is not specified, the DEFAULT CLIENT_AUTH section is used.

See Section 14.4 for further discussion and examples of use of these channel keywords.

2.3.4.44 Use authenticated address from SMTP AUTH in header (authrewrite)

The authrewrite channel keyword may be used on a source channel to have PMDF propagate authenticated originator information, if available, into the headers. Normally the SMTP AUTH information is used, though this may be overriden via the FROM_ACCESS mapping; see Section 16.1.3. authrewrite takes a required integer value, according to the following table:

Value	Usage
1	Add a Sender: header, or a Resent-sender: header if a Resent-from: or Resent-sender: was already present, containing the AUTH originator
2	Add a Sender: header containing the AUTH originator

2.3.4.45 Transport Layer Security (maytls, maytlsclient, maystlsserver, musttls, musttlsclient, musttlsserver, notls, notlsclient, notlsserver, tlsswitchchannel)

Note: These channel keywords are only supported for PMDF-TLS sites.

The maytls, maytlsclient, maytlsserver, musttls, musttlsclient, musttlsserver, notls, notlsclient, notlsserver, and tlsswitchchannel channel keywords are used to configure TLS use during the SMTP protocol by SMTP based channels such as TCP/IP channels. notls is the default, and means that TLS will not be permitted or attempted. It subsumes the notlsclient keyword, which means that TLS use will not be attempted by the PMDF SMTP client on outgoing connections (the STARTTLS command will not be issued during outgoing connections) and the notlsserver keyword, which means that TLS use will not be permitted by the PMDF SMTP server on incoming connections (the STARTTLS extension will not be advertised by the SMTP server nor the command itself accepted). Specifying maytls causes PMDF to offer TLS to incoming connections and to attempt TLS upon outgoing connections. It subsumes maytlsclient, which means that the PMDF SMTP client will attempt TLS use when sending outgoing messages, if sending to an SMTP server that supports TLS, and maytlsserver, which means that the PMDF SMTP server will advertise support for the STARTTLS extension and will allow TLS use when receiving messages. Specifying musttls will cause PMDF to insist upon TLS in both outgoing and incoming connections; e-mail will not be exchanged with remote systems that fail to successfully negotiate TLS use. It subsumes musttlsclient, which means that the PMDF SMTP client will insist on TLS use when sending outgoing messages and will not send to SMTP servers that

do not successfully negotiate TLS use (PMDF will issue the STARTTLS command and that command must succeed), and `musttlsserver`, which means that the PMDF SMTP server will advertise support for the STARTTLS extension and will insist upon TLS use when receiving incoming messages and will not accept messages from clients that do not successfully negotiate TLS use. The `tlsswitchchannel` keyword is used to cause incoming connections to be switched to a specified channel upon a client's successful TLS negotiation. It takes a required value, specifying the channel to which to switch.

See Chapter 15 for additional discussion of TLS.

2.3.4.46 MS Exchange gateway channels (`msexchange`, `nomsexchange`)

The `msexchange` channel keyword may be used on TCP/IP channels to tell PMDF that this is a channel that communicates with MS Exchange gateways and clients. When placed on an incoming TCP/IP channel which has SASL enabled (via a `maysaslserver` or `mustsaslserver` keyword), it causes PMDF's SMTP server to advertise AUTH using an "incorrect" format (based upon the original ESMTP AUTH specification, which was actually incompatible with correct ESMTP usage, rather than the newer, corrected AUTH specification). Some MS Exchange clients, for instance, will not recognize the correct AUTH format and will only recognize the incorrect AUTH format.

The `msexchange` channel keyword also causes advertisement (and recognition) of broken TLS commands.

`nomsexchange` is the default.

2.3.4.47 Host name to use when correcting incomplete addresses (`remotehost`, `noremotehost`, `defaulthost`, `nodefaulthost`)

PMDF often receives from misconfigured or incompliant mailers and SMTP clients addresses which do not contain a domain name. PMDF, showing at least some respect for standards, must attempt to make such addresses legal before allowing them to pass further. PMDF does this by appending a domain name to the address (*e.g.,* appends "@example.com" to "mrochek"). For envelope To: addresses missing a domain name, PMDF always assumes that the local host name should be appended. However for other addresses, such as From: addresses, in the case of the PMDF SMTP server there are at least two reasonable choices for the domain name: the local PMDF host name and the remote host name reported by the client SMTP. Or in some cases, there may be yet a third reasonable choice—a particular domain name to add to messages coming in that channel. Now, either of these two first choices are likely to be correct as both may occur operationally with some frequency. The use of the remote host's domain name is appropriate when dealing with improperly configured SMTP clients. The use of the local host's domain name may be appropriate when dealing with a lightweight remote mail client such as a POP or IMAP client that uses SMTP to post messages. Or if lightweight remote mail clients such as POP or IMAP clients "ought" to have their own specific domain name which is not that of the local host, then adding that specific other domain name may be appropriate. The best that PMDF can do is to allow the choice to be made on a channel by channel basis.

The remotehost channel keyword specifies that the remote host's name should be used. The noremotehost channel keyword specifies that the local host's name should be used. noremotehost is the default.

The defaulthost channel keyword is used to specify a particular host name to append to incoming bare usernames; it must to be followed by the domain name to use in completing addresses coming in that channel. nodefaulthost is the default.

Note that the switchchannel keyword described above can be used to associated incoming SMTP connections with a particular channel. This facility can be used to group remote mail clients on a channel where they will receive proper treatment. Alternatively, it is an unconditionally simpler proposition to deploy standards-compliant remote mail clients (even if a multitude of incompliant clients are in use) rather than attempting to fix the network-wide problem on your PMDF hosts.

2.3.4.48 Normalizing messages that lack any recipient headers (missingrecipientpolicy)

According to RFC 822, messages are required to contain at least one recipient header: a To:, Cc:, or Bcc: header. This RFC states that a message without any such headers is illegal. This requirement has been relaxed in the updated RFC 2822 standard: such messages are no longer illegal. However, some remote systems that conform to RFC 822 will not accept these messages. In many cases, it can be useful to have PMDF modify the message to include at least one recipient header.

The missingrecipientpolicy channel keyword takes an integer value specifying what approach to use for such messages; the default value, if the keyword is not explicitly present, is 1, meaning that no action is taken.

Value	Action
1	Pass the message through unchanged
2	Place envelope To: recipients in a To: header
3	Place all envelope To: recipients in a single Bcc: header
4	Generate an empty group construct To: header (i.e. "To: Recipients not specified: ;")
5	Generate a blank Bcc: header
6	Reject the message

Note that the MISSING_RECIPIENT_POLICY PMDF option, discussed in Section 7.3.1, can be used to set a PMDF system default for this sort of behavior.

2.3.4.49 Strip illegal blank recipient headers (dropblank)

In RFC 822 (Internet) messages, any To:, Resent-To:, Cc:, or Resent-Cc: header is required to contain at least one address—such a header may not have a blank value. Nevertheless, some mailers may emit such illegal headers. The dropblank channel keyword, if specified on a source channel, causes PMDF to strip any such illegal blank headers from incoming messages.

2.3.4.50 Eight bit capability (eightbit, eightnegotiate, eightstrict, sevenbit)

Some transports restrict the use of characters with ordinal values greater than 127 (decimal). Most notably, some SMTP servers will strip the high bit and thus garble messages that use characters in this "eight bit" range.[g]

PMDF provides facilities to automatically encode such messages so that troublesome eight bit characters do not appear directly in the message. This encoding can be applied to all messages on a given channel by specifying the sevenbit keyword. A channel should be marked eightbit if no such restriction exists.

Some transports such as extended SMTP may actually support a form of negotiation to determine if eight bit characters can be transmitted. The eightnegotiate keyword can be used to instruct the channel to encode messages when negotiation fails. This is the default for all channels; channels that do not support negotiation will simply assume that the transport is capable of handling eight bit data.

The eightstrict keyword tells PMDF to reject any messages that contain unnegotiated eight bit data.

2.3.4.51 Automatic character set labelling (charset7, charset8, charsetesc)

The MIME specification provides a mechanism to label the character set used in a plain text message. Specifically, a "charset=" parameter can be specified as part of the Content-type: header line. Various character set names are defined in MIME, including US-ASCII (the default), ISO-8859-1, ISO-8859-2, and so on. Additional character set names will undoubtedly be added to the list in the future.

Most existing systems and user agents, however, do not provide any mechanism for generating these character set labels. In particular, plain text messages sent from VMS MAIL are not properly labelled. The charset7, charset8, and charsetesc channel keywords provide a per-channel mechanism to specify character set names to be inserted into message headers. Each keyword requires a single argument giving the character set name. The names are not checked for validity. Note, however, that character set conversion can only be done on character sets specified in the PMDF character set definition file charsets.txt found in the PMDF table directory, (*i.e.*, PMDF_TABLE:charsets.txt on OpenVMS or /pmdf/table/charsets.txt on UNIX. The names defined in this file should be used if possible.

[g] Indeed, there have even been reports of SMTP servers which will crash when presented with eight bit data.

The charset7 character set name is used if the message contains only seven bit characters; the charset8 will be used if eight bit data is found in the message; charsetesc will be used if a message containing only seven bit data happens to contain the escape character. If the appropriate keyword is not specified no character set name will be inserted into the Content-type: header lines.

These character set specifications never override existing labels; that is, they have no effect if a message already has a character set label or is of a type other than text.

It is usually appropriate to label the PMDF local channel as follows:

```
l ... charset7 US-ASCII charset8 ISO-8859-1 ...
official-host-name
```

OpenVMS systems actually use the DEC Multinational Character Set (DEC-MCS). The character set is very close to ISO-8859-1, however, so this labelling will work well enough in most cases. If absolute accuracy is an issue, the local channel can be marked as using DEC-MCS

```
l ... charset7 US-ASCII charset8 DEC-MCS ...
official-host-name
```

and an appropriate character set conversion can be set up to convert DEC-MCS to ISO-8859-1 as needed. See Chapter 6 for details on how to set up such conversions.

The charsetesc keyword tends to be particularly useful on channels that receive unlabelled messages using Japanese or Korean character sets that contain the escape character.

2.3.4.52 Restrictions on message line lengths (linelength)

The SMTP specification allows for lines of text containing up to 1000 bytes. However, some transports may impose more severe restrictions on line length. For example, while the MAIL-11 transport over DECnet supports lines of any length, the access to this transport provided by the VMS MAIL foreign protocol interface limits lines to 255 characters or less.

The linelength keyword provides a mechanism for limiting the maximum permissible message line length on a channel by channel basis. Messages queued to a given channel with lines longer than the limit specified for that channel will be encoded automatically. The various encodings available in PMDF always result in a reduction of line length to fewer than 80 characters. The original message may be recovered after such encoding is done by applying an appropriate decoding filter. (In many cases PMDF MAIL and other MIME-aware user agents are able to detect that such decoding is necessary and perform it automatically.)

Note that encoding can only reduce line lengths to fewer than 80 characters. For this reason specification of line length values less than 80 may not actually produce lines with lengths that comply with the stated restriction.

Note also that `linelength` causes encoding of data so as to do "soft" line wrapping for transport purposes. The encoding is normally decoded at the receiving side so that the original "long" lines are recovered. For "hard" line wrapping, see instead the "Record,Text" CHARSET-CONVERSION; Chapter 6.

2.3.4.53 Delivering foreign format messages to VMS MAIL (OpenVMS) (`foreign`, `noforeign`)

PMDF detects encoded messages automatically and is capable of decoding and delivering them as foreign format messages to VMS MAIL. Automatic decoding saves time since the user no longer needs to extract the message, edit it, and run it through a utility such as PMDF DECODE to obtain the contents.

The `foreign` channel keyword enables this processing; this keyword is only effective for OpenVMS systems and only on the local (l) channel, the DECnet MAIL-11 channel (d), and mail_ channels. The `noforeign` channel keyword disables the use of foreign format messages, and is the default for all channels. Note that this default represents a change from the default behavior of versions of PMDF prior to V5.1.

2.3.4.54 Conversion of application/octet-stream material (`convert_octet_stream`, `noconvert_octet_stream`)

MIME provides a general-purpose type for exchange of pure untyped binary data. Such data may or may not be usable in any given circumstance; no other information about the data is available. Various PMDF channels provide mechanisms for dealing with such data that may or may not be appropriate. The `convert_octet_stream` and `noconvert_octet_stream` keywords control these mechanisms; if the former is specified conversions are performed and if the latter is specified no conversions are performed. The latter keyword is the default for all channels.

Optional channel-specific conversions available include:

1. The OpenVMS L, DECnet MAIL-11, and mail_ channels all can optionally convert application/octet-stream data into VMS MAIL foreign messages. See also the `foreign` keyword, Section 2.3.4.53.

2.3.4.55 Channel-specific use of the reverse database (`reverse`, `noreverse`)

The `reverse` keyword tells PMDF that addresses in messages queued to the channel should be checked against and possibly modified by the address reversal database or REVERSE mapping if either exists. `noreverse` exempts addresses in messages queued to the channel from address reversal processing. The `reverse` keyword is the default. See Section 3.3.2 information about address reversal.

2.3.4.56 Inner header rewriting (noinner, inner)

PMDF only interprets the contents of header lines when necessary. However, MIME messages can contain multiple sets of message headers as a result of the ability to imbed messages within messages (message/rfc822). PMDF normally only interprets and rewrites the outermost set of message headers. PMDF can optionally be told to apply header rewriting to inner headers within the message as well.

This behavior is controlled by the use of the `noinner` and `inner` keywords. `noinner` tells PMDF not to rewrite inner message header lines. It is the default. `inner` tells PMDF to parse messages and rewrite inner headers.

These keywords can be applied to any channel.

2.3.4.57 Restricted mailbox encoding (restricted, unrestricted)

Some mail systems have great difficulty dealing with the full spectrum of addresses allowed by RFC 822. A particularly common example of this is sendmail-based mailers with incorrect configuration files. Quoted local-parts (or mailbox specifications) are a frequent source of trouble:

```
"alonso, king"@naples.example.com
```

This is such a major source of difficulty that a methodology was laid out in RFC 1137 to work around the problem. The basic approach is to remove quoting from the address and then apply a translation that maps the characters requiring quoting into characters allowed in an atom (see RFC 822 for a definition of an atom as it is used here). For example, the preceeding address would become:

```
alonso#m#_king@naples.example.com
```

The `restricted` channel keyword tells PMDF that the channel connects to mail systems that require this encoding. PMDF then encodes quoted local-parts in both header and envelope addresses as messages are written to the channel. Incoming addresses on the channel are decoded automatically.

The `unrestricted` keyword tells PMDF not to perform RFC 1137 encoding and decoding. `unrestricted` is the default.

Note: The `restricted` keyword should be applied to the channel that connects to systems unable to accept quoted local-parts. It should *not* be applied to the channels that actually generate the quoted local-parts! (It is assumed that a channel capable of generating such an address is also capable of handling such an address.)

2.3.4.58 Additional message header lines in VMS MAIL (headerbottom, headerinc, headeromit)

VMS MAIL only provides support for four message header lines: From:, To:, Cc: and Subject:. However, RFC 822 headers can contain many additional types of header lines. On OpenVMS systems, PMDF supports these additional header lines by optionally prepending or appending them to the message body whenever a message is delivered to a local user.

This behavior is controlled by the use of the headerinc, headeromit, and headerbottom keywords in the local channel block. The default is headerinc, which tells PMDF to prepend the header lines to the message — this is the only one of these keywords supported on UNIX systems. On OpenVMS systems, the keywords headerbottom, which tells PMDF to append the header lines to the end of the message, and headeromit, which tells PMDF to strip all header lines, are also available.

Extreme care should be taken not to use these keywords on channels connecting to other message handling systems — relocating or eliminating message headers violates RFC 821 and RFC 822 and can lead to serious problems. Note also that many seemingly "bothersome" header lines may contain valuable information required to decode messages, track down problems, or even authenticate who really sent the message and thus thwart attempts at forging mail messages.

The headerinc, headeromit, and headerbottom keywords will have no effect if they are specified in any channel block that does not use VMS MAIL or SMTP as its delivery mechanism, and indeed the headeromit and headerbottom keywords are only available on the OpenVMS version of PMDF. Their usage with SMTP channels should be limited to special SMTP channels; they should never be used with a TCP/IP channel such as tcp_local, *etc.*. Their usage with SMTP channels is intended for delivery of messages to local users on, for instance, microcomputers.

2.3.4.59 Trimming message header lines (headertrim, noheadertrim, headerread, noheaderread, innertrim, noinnertrim)

PMDF provides per-channel facilities for trimming or removing selected message header lines from messages. This is done through a combination of a channel keyword and an associated header option file or two. The headertrim keyword instructs PMDF to consult a header option file associated with the channel and to trim the headers on messages queued to that destination channel accordingly, *after the original message headers are processed*. The noheadertrim keyword bypasses header trimming. noheadertrim is the default.

The innertrim keyword instructs PMDF to perform header trimming on inner message parts, *i.e.*, embedded MESSAGE/RFC822 parts, as well. The noinnertrim keyword, which is the default, tells PMDF not to perform any headertrimming on inner message parts.

The headerread keyword instructs PMDF to consult a header option file associated with the channel and to trim the headers on messages enqueued by that source channel accordingly, *before the original message headers are processed*. Note that headertrim header trimming, on the other hand, is applied after the messages have been processed,

and is destination channel, rather than source channel, related. The noheaderread keyword bypasses message enqueue header trimming. noheaderread is the default.

Unlike the headeromit and headerbottom keywords, the headertrim and headerread keywords may be applied to any channel whatsoever. Note, however, that stripping away vital header information from messages may cause improper operation of PMDF. Be extremely careful when selecting headers to remove or limit. This facility exists because there are occassional situations where selected header lines must be removed or otherwise limited. *Do not merely trim header lines away because you or your users find them annoying — those header lines are there for a reason. More often than not, the header lines that users feel are superfluous are among the most important. Before trimming or removing any header line, be sure that you understand the usage of that header line and have considered the possible implications of its removal. Consult RFC 822, a copy of which may be found in the RFC subdirectory of the PMDF documentation directory.*[h]

Header options files for the headertrim and innertrim keywords have names of the form *channel*_headers.opt with *channel* the name of the channel with which the header option file is associated. Similarly, header options files for the headerread keyword have names of the form *channel*_read_headers.opt. These files are stored in the PMDF table directory, PMDF_TABLE: on OpenVMS or /pmdf/table/ on UNIX. See Section 2.3.7 for information on the format of these files.

2.3.4.60 Encoding header (ignoreencoding, ignoremessageencoding, ignoremultipartencoding, interpretencoding, interpretmessageencoding, interpretmultipartencoding)

PMDF has the ability to convert various non-standard message formats to MIME via the Yes CHARSET-CONVERSION; see Chapter 6. In particular, the RFC 1154 format uses a non-standard Encoding: header. However, some gateways emit incorrect information on this header line, with the result that sometimes it is desirable to ignore this header.

The ignoreencoding keyword instructs PMDF to ignore any Encoding: header. (Note that unless PMDF has a CHARSET-CONVERSION enabled, such headers will be ignored in any case.) Similarly, the ignoremessageencoding keyword instructs PMDF to ignore any Encoding: header in embedded messages. And the ignoremultipartencoding keyword instructs PMDF to ignore any Encoding: header in multipart messages.

The interpretencoding keyword instructs PMDF to pay attention to any Encoding: header, if otherwise configured to do so, and is the default. Similarly, the interpretmessageencoding keyword instructs PMDF to pay attention to any Encoding: header in embedded messages. And the interpretmultipartencoding keyword instructs PMDF to pay attention to any Encoding: header in multipart messages.

[h] PMDF_DOC:[rfc] on OpenVMS; /pmdf/doc/rfc/ on UNIX.

2.3.4.61 Generation of X-Envelope-to: header lines (x_env_to, nox_env_to)

The x_env_to and nox_env_to keywords control the generation or suppression of X-Envelope-to: header lines on copies of messages queued to a specific channel. On channels that are marked with the single keyword, the x_env_to keyword enables generation of these headers while the nox_env_to will remove such headers from enqueued messages. The default is nox_env_to; (note that this default behavior is a change in PMDF V5.0 from previous versions of PMDF). Note that the fact that x_env_to also requires the single keyword in order to take effect represents a change of behavior from PMDF V5.1 and earlier. See Section 19.1.4.14 for a discussion of the meaning of X-Envelope-to: headers.

2.3.4.62 Envelope to address in Received: header (receivedfor, noreceivedfor, receivedfrom, noreceivedfrom)

The receivedfor keyword instructs PMDF that if a message is addressed to just one envelope recipient, to include that envelope To: address in the Received: header it constructs. receivedfor is the default. The noreceivedfor keyword instructs PMDF to construct Received: headers without including any envelope addressee information.

The receivedfrom keyword instructs PMDF to include the original envelope From: address when constructing a Received: header for an incoming message if PMDF has changed the envelope From: address due to, for instance, certain sorts of mailing list expansions. receivedfrom is the default. The noreceivedfrom keyword instructs PMDF to construct Received: headers without including the original envelope From: address.

2.3.4.63 Postmaster address (aliaspostmaster, returnaddress, noreturnaddress, returnpersonal, noreturnpersonal)

By default, the Postmaster return address used when PMDF constructs bounce or notification messages is postmaster@*local-host*, where *local-host* is the official local host name (the name on the local channel), and the Postmaster personal name is "PMDF e-Mail Interconnect". Care should be taken in the selection of the Postmaster address—an illegal selection may cause rapid message looping and pile-ups of huge numbers of spurious error messages.

The RETURN_ADDRESS and RETURN_PERSONAL PMDF options can be used to set a PMDF system default for the Postmaster address and personal name. Or if per channel controls are desired, the returnaddress and returnpersonal channel keywords may be used. returnaddress and returnpersonal each take a required argument specifying the Postmaster address and Postmaster personal name, respectively. noreturnaddress and noreturnpersonal are the defaults and mean to use the default values, either defaults established via the RETURN_ADDRESS and RETURN_PERSONAL options, or the normal default values if such options are not set.

If the aliaspostmaster keyword is placed on a channel, then any messages addressed to the username "postmaster" (lowercase, uppercase, or mixed case) at the official channel name will be redirected to postmaster@*local-host*, where *local-host* is the official local host name (the name on the local channel). Note that Internet

standards require that any domain in the DNS that accepts mail have a valid postmaster account that will receive mail. So this keyword can be useful when it is wanted to centralize postmaster responsibilities, rather than having separate postmaster accounts for separate domains. That is, whereas `returnaddress` controls what return postmaster address is used when PMDF generates a notification message *from* the postmaster, `aliaspostmaster` affects what PMDF does with messages addressed *to* the postmaster.

2.3.4.64 Blank envelope return addresses (`returnenvelope`)

The `returnenvelope` keyword takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether or not return notifications generated by PMDF are written with a blank envelope address or with the address of the local postmaster. Setting the bit forces the use of the local postmaster address, clearing the bit forces the use of a blank addresses. Note that the use of a blank address is mandated by RFC 1123. However, some systems do not handle blank envelope from address properly and may require the use of this option.

Bit 1 (value = 2) controls whether or not PMDF replaces all blank envelope addresses with the address of the local postmaster. Again, this is used to accomodate incompliant systems that don't conform to RFC 821, RFC 822, or RFC 1123.

Note that the RETURN_ENVELOPE PMDF option can be used to set a PMDF system default for this sort of behavior.

2.3.4.65 Mapping Reply-to: header (`usereplyto`)

The `usereplyto` keyword controls the mapping of the Reply-to: header. The default is `usereplyto 0`, which means to use the channel default behavior (which varies from channel to channel).

Value	Action
-1	Never map Reply-to: addresses to anything.
0	Use the channel default mapping of Reply-to: addresses; (varies from channel to channel). This is the default.
1	Map Reply-to: to From: if no usable From: address exists.
2	If there is a usable Reply-to: address, then map it to From:; otherwise fall back to the From: address.

2.3.4.66 Mapping Resent- headers when gatewaying to non RFC 822 environments (`useresent`)

The `useresent` keyword controls the use of Resent- headers when gatewaying to environments that do not support RFC 822 headers. This keyword takes a single integer-valued argument. Legal values include:

Value	Action
+2	Use any Resent- headers that are present to generate address information.
+1	Only use Resent-From: headers to generate address information; all other Resent- headers are ignored.
0	Do not use Resent- headers to generate address information. This is the default.

Currently the `useresent` keyword applies for the l (lowercase "L") channel on OpenVMS, and for PMDF-MR, PMDF-X400, and some PMDF-LAN channels.

Note that the default of 0 constitutes a change in the behavior of the OpenVMS l channel compared with PMDF version 4.3 and earlier.

2.3.4.67 Comments in address message headers (`commentinc`, `commentomit`, `commentstrip`, `commenttotal`, `sourcecommentinc`, `sourcecommentomit`, `sourcecommentstrip`, `sourcecommenttotal`)

PMDF only interprets the contents of header lines when necessary. However, all registered headers containing addresses must be parsed in order to rewrite and eliminate shortform addresses and otherwise convert them to legal addresses. During this process comments (strings enclosed in parentheses) are extracted and may optionally be modified or excluded when the header line is rebuilt.

On destination channels, this behavior is controlled by the use of the `commentinc`, `commentomit`, `commentstrip`, and `commenttotal` keywords. `commentinc` tells PMDF to retain comments in header lines. It is the default. `commentomit` tells PMDF to remove any comments from addressing headers, *e.g.*, To:, From:, Cc: headers, *etc*. `commenttotal` tells PMDF to remove any comments from all headers, except Received: headers; as such, this keyword is not normally useful or recommended. And finally, `commentstrip` tells PMDF to strip any nonatomic characters from all comment fields.

On source channels, this behavior is controlled by the use of the `sourcecommentinc`, `sourcecommentomit`, `sourcecommentstrip`, and `sourcecommenttotal` keywords. `sourcecommentinc` tells PMDF to retain comments in header lines. It is the default. `sourcecommentomit` tells PMDF to remove any comments from addressing headers, *e.g.*, To:, From:, Cc: headers, *etc*. `sourcecommenttotal` tells PMDF to remove any comments from all headers, except Received: headers; as such, this keyword is not normally useful or recommended. And finally, `sourcecommentstrip` tells PMDF to strip any nonatomic characters from all comment fields.

These keywords can be applied to any channel.

2.3.4.68 Personal names in address message headers (personalinc, personalomit, personalstrip, sourcepersonalinc, sourcepersonalomit, sourcepersonalstrip)

PMDF only interprets the contents of header lines when necessary. However, all registered headers containing addresses must be parsed in order to rewrite and eliminate shortform addresses and otherwise convert them to legal addresses. During this process personal names (strings preceeding angle-bracket-delimited addresses) are extracted and may optionally be modified or excluded when the header line is rebuilt.

On destination channels, this behavior is controlled by the use of the personalinc, personalomit, and personalstrip keywords. personalinc tells PMDF to retain personal names in the headers. It is the default. personalomit tells PMDF to remove all personal names. And finally, personalstrip tells PMDF to strip any nonatomic characters from all personal name fields.

On source channels, this behavior is controlled by the use of a sourcepersonalinc, sourcepersonalomit, or sourcepersonalstrip keyword. sourcepersonalinc tells PMDF to retain personal names in the headers. It is the default. sourcepersonalomit tells PMDF to remove all personal names. And finally, sourcepersonalstrip tells PMDF to strip any nonatomic characters from all personal name fields.

These keywords can be applied to any channel.

2.3.4.69 Alias file and alias database probes (aliaslocal)

Normally only addresses rewritten to the local channel (that is, the l channel on OpenVMS and UNIX) are looked up in the alias file and alias database. The aliaslocal keyword may be placed on a channel to cause addresses rewritten to that channel to be looked up in the alias file and alias database also. The exact form of the lookup probes that will be made is then controlled by the ALIAS_DOMAINS PMDF option; see Section 7.3.1.

2.3.4.70 Validating local part of address (validatelocalnone, validatelocalsystem, validatelocalmsgstore)

The validatelocalnone, validatelocalsystem, and validatelocalmsgstore channel keywords control whether any validity check on the local part (username) of an address is performed when messages are enqueued to the channel. Different sorts of channels have different defaults; most channels default to validatelocalnone, meaning that no validation of the local part of the address is performed by the channel doing the enqueuing to the channel in question.

The local channel defaults to validatelocalsystem, meaning that the local part (username) of an address must be a valid, e-mail receiving account on the system. More specifically, validatelocalsystem means that on UNIX platforms, the local part (username) must have an account on the system, or on OpenVMS platforms that the local part (username) must have an account or VMS MAIL profile entry.

The msgstore channel defaults to validatelocalmsgstore, meaning that the local part (username) of an address must be a valid MessageStore or popstore account, or have an entry in the MessageStore forward database, and has not been marked DISMAIL or be over quota (if REJECT_OVER_QUOTA is set).

When validatelocalnone is on a channel, messages matching that channel are enqueued to the channel with no validation by the enqueuing channel; it will be up to the destination channel itself to validate the address. So for instance if validatelocalnone were placed on the local channel, then incoming SMTP messages apparently matching the local channel would be accepted by the SMTP server and enqueued to the local channel; if the local part turned out not to be a valid account, that would not be discovered until the local channel itself actually ran and checked the local part.

Conversely, if the name space for some other destination channel, say a MRIF_A1 channel, happened to exactly match the name space for the accounts on the local channel, then placing validatelocalsystem on the MRIF_A1 channel would cause enqueuing PMDF agents such as the SMTP server to reject messages destined for the MRIF_A1 channel for which the local part (username) could not be validated as if it were a VMS MAIL account.

2.3.4.71 Subaddresses (subaddressexact, subaddressrelaxed, subaddresswild)

As background regarding the concept of subaddresses, the PMDF local and msgstore channels interpret a + character in the local portion of an address (the mailbox portion) specially: in an address of the form *name*+*subaddress*@*domain* PMDF considers the portion of the mailbox after the plus character a *subaddress*. The msgstore channel when delivering to a popstore account and the local channel treat a subaddress as additional cosmetic information and actually deliver to the account *name*, without regard to the subaddress; the msgstore channel when delivering to a PMDF MessageStore account interprets the subaddress as the folder name to which to deliver.

Subaddresses also affect the lookup of aliases by the local channel (that is, the local channel on OpenVMS or UNIX) and the lookup of aliases by any channel marked with the aliaslocal keyword, and the lookup of mailboxes by the directory channel. The exact handling of subaddresses for such matching is configurable: when comparing an address against an entry, PMDF always first checks the entire mailbox including the subaddress for an exact match; whether or not PMDF performs additional checks after that is configurable.

Note that the msgstore channel behaviors of ignoring subaddresses as far as actually delivering to a popstore account and interpreting subaddresses as folder names when delivering to a MessageStore account are true on all platforms. On all platforms, you must use the aliaslocal keyword on the msgstore channel if you want to compare addresses against the alias file and alias database. Hence the subaddress* keywords are relevant only if the msgstore channel is marked with the aliaslocal keyword.

The subaddressexact keyword instructs PMDF to perform no special subaddress handling during entry matching; the entire mailbox, including the subaddress, must match an entry in order for the alias to be considered to match. No additional comparisons (in particular, no wildcarded comparisons or comparisons with the subaddress removed) will be performed. The subaddresswild keyword instructs PMDF that after

looking for an exact match including the entire subaddress, PMDF should next look for an entry of the form *name*+*. The subaddressrelaxed keyword instructs PMDF that after looking for an exact match and then a match of the form *name*+*, that PMDF should make one additional check for a match on just the *name* portion. With subaddressrelaxed, an alias entry of the form

```
name:   newname+*
```

will match either name or name+*subaddress*, transforming a plain name to newname, and transforming name+*subaddress* to newname+*subaddress*. subaddressrelaxed is the default.

Thus the subaddresswild keyword or the subaddressrelaxed keyword may be useful when aliases or a directory channel are in use yet users want to receive mail addressed using arbitrary subaddresses. These keywords obviate the need for a separate entry for every single subaddress variant on an address.

Note that these keywords only make sense for the local channel (that is, the l channel on OpenVMS or UNIX) and the directory channel, or any channel marked with the aliaslocal keyword.

2.3.4.72 Two or four digit date conversion (datefour, datetwo)

The original RFC 822 specification called for two digit years in the date fields in message headers. This was later changed to four digits by RFC 1123. However, some older mail systems cannot accommodate four digit dates. In addition, some newer mail systems can no longer tolerate two digit dates! (Please note that systems which cannot handle both formats are in violation of the standards.)

The datefour and datetwo keywords control PMDF's processing of the year field in message header dates. datefour, the default, instructs PMDF to expand all year fields to four digits. Two digit dates with a value less than 50 will have 2000 added while values greater than 50 will have 1900 added.

datetwo instructs PMDF to remove the leading two digits from four digit dates. This is intended to provide compatibility with incompliant mail systems that require two digit dates; it should never be used for any other purpose.

2.3.4.73 Day of week in date specifications (dayofweek, nodayofweek)

The RFC 822 specification allows for a leading day of the week specification in the date fields in message headers. However, some systems cannot accomodate day of the week information. This makes some systems reluctant to include this information, even though it is quite useful information to have in the headers.

The dayofweek and nodayofweek keywords control PMDF's processing of day of the week information. dayofweek, the default, instructs PMDF to retain any day of the week information and to add this information to date/time headers if it is missing.

nodayofweek instructs PMDF to remove any leading day of the week information from date/time headers. This is intended to provide compatibility with incompliant mail systems that cannot process this information properly; it should never be used for any other purpose.

2.3.4.74 Automatic splitting of long header lines (maxheaderaddrs, maxheaderchars)

Some message transports, notably some sendmail implementations, cannot process long header lines properly. This often leads not just to damaged headers but to erroneous message rejection. Although this is a gross violation of standards it is nevertheless a common problem.

PMDF provides per-channel facilities to split (break) long header lines into multiple, independent header lines. The maxheaderaddrs keyword controls how many addresses can appear on a single line. The maxheaderchars keyword controls how many characters can appear on a single line. Both keywords require a single integer parameter that specifies the associated limit. By default, no limit is imposed on the length of a header line nor on the number of addresses which may appear.

2.3.4.75 Header alignment and folding (headerlabelalign, headerlinelength)

The headerlabelalign keyword controls the alignment point for message headers enqueued on this channel; it takes an integer-valued argument. The alignment point is the margin where the contents of headers are aligned. For example, sample headers with an alignment point of 10 would appear as follows:

```
To:      ariel@example.com
From:    caliban@example.com
Subject: Alignment test
```

The default headerlabelalign is 0, which causes headers not to be aligned.

The headerlinelength keyword controls the length of message header lines enqueued on this channel. The default, if this keyword is not explicitly set, is 80. Lines longer than this are folded in accordance with RFC 822 folding rules.

Note that these keywords only control the format of the headers of the message in the message queue; the actual display of headers is normally controlled by the user agent. In addition, headers are routinely reformatted as they are transported across the Internet, so these keywords may have no visible effect even when used in conjunction with simple user agents that do not reformat message headers.

2.3.4.76 Automatic defragmentation of message/partial messages (defragment, nodefragment)

The MIME standard provides the message/partial content type for breaking up messages into smaller parts. This is useful when messages have to traverse networks with size limits. Information is included in each part so that the message can be automatically reassembled once it arrives at its destination.

The defragment channel keyword and the defragmentation channel provide the means to reassemble messages in PMDF. When a channel is marked defragment any message/partial messages queued to the channel will be placed in the defragmentation channel queue instead. Once all the parts have arrived the message is rebuilt and sent on its way.

The nodefragment disables this special processing. nodefragment is the default.

A defragment channel must be added to the PMDF configuration file in order for the defragment keyword to have any effect. If your configuration was built by the PMDF configuration utility, then you should already have such a channel. If not consult Section 26.3.

2.3.4.77 Automatic fragmentation of large messages (maxblocks, maxlines)

Some mail systems or network transports cannot handle messages that exceed certain size limits. PMDF provides facilities to impose such limits on a channel-by-channel basis. Messages larger than the set limits will automatically be split (fragmented) into multiple, smaller messages. The Content-type: used for such fragments is message/partial, and a unique id parameter is added so that parts of the same message can be associated with one another and, possibly, be automatically reassembled by the receiving mailer.

Message fragmentation and defragmentation may also be used to effectively provide "checkpointing" of message transmission.

The maxblocks and maxlines keywords are used to impose size limits beyond which automatic fragmentation will be activated. Both of these keywords must be followed by a single integer value. maxblocks specifies the maximum number of blocks allowed in a message. A PMDF block is normally 1024 bytes; this can be changed with the BLOCK_SIZE option in the PMDF option file; see Section 7.3.5. maxlines specifies the maximum number of lines allowed in a message. These two limits can be imposed simultaneously if necessary.

Message headers are to a certain extent included in the size of a message. Since message headers cannot be split into multiple messages, and yet they themselves may exceed the specified size limits, a rather complex mechanism is used to account for message header sizes. This logic is controlled by the MAX_HEADER_BLOCK_USE and MAX_HEADER_LINE_USE options in the PMDF option file.

MAX_HEADER_BLOCK_USE is used to specify a real number between 0 and 1. The default value is 0.5. A message's header is allowed to occupy this much of the total number of blocks a message can consume (specified by the maxblocks keyword). If the message header is larger, PMDF takes the product of MAX_HEADER_BLOCK_USE and

maxblocks as the size of the header; *i.e.,* the header size is taken to be the smaller of the actual header size and maxblocks * MAX_HEADER_BLOCK_USE.

For example, if maxblocks is 10 and MAX_HEADER_BLOCK_USE is the default, 0.5, any message header that is larger than 5 blocks is treated as a 5 block header, and if the message is 5 or fewer blocks in size it will not be fragmented. A value of 0 will cause headers to be effectively ignored insofar as message size limits are concerned. A value of 1 allows headers to use up all of the size that's available. Note, however, that each fragment will always contain at least one message line, regardless of whether or not the limits are exceeded by this.

MAX_HEADER_LINE_USE operates in a similar fashion in conjunction with the maxlines keyword.

2.3.4.78 Absolute message size limits (blocklimit, noblocklimit, linelimit, nolinelimit, sourceblocklimit)

Although fragmentation may be used to break messages into smaller pieces automatically, it may also be appropriate in some cases to simply reject outright messages larger than some administratively defined limit, (*e.g.,* so as to avoid service denial attacks).

The blocklimit, linelimit and sourceblocklimit keywords are used to impose absolute size limits. Each of these keywords must be followed by a single integer value. blocklimit specifies the maximum number of blocks allowed in a message. PMDF will reject attempts to queue messages containing more blocks than this to the channel. The sourceblocklimit specifies the maximum number of blocks allowed in an incoming message. PMDF will reject attempts to submit a message containing more blocks than this to the channel. In other words, blocklimit applies to destination channels; sourceblocklimit applies to source channels. A PMDF block is normally 1024 bytes; this can be changed with the BLOCK_SIZE option in the PMDF option file. linelimit specifies the maximum number of lines allowed in a message. Note that linelimit counts both header lines and body lines of a message. PMDF will reject attempts to queue messages containing more than this number of lines to the channel. These limits can be imposed simultaneously if necessary.

Note that the PMDF options LINE_LIMIT and BLOCK_LIMIT can be used to impose similar limits on all channels. These limits have the advantage that since they apply across all channels PMDF's servers can make them known to mail clients prior to obtaining message recipient information. This simplifies the process of message rejection in some protocols.

The nolinelimit and noblocklimit channel keywords are the default and mean that no limits are imposed, other than any global limits imposed via the LINE_LIMIT or BLOCK_LIMIT PMDF options.

2.3.4.79 Specify maximum length header that PMDF will rewrite (maxprocchars)

Processing of long header lines containing lots of addresses can consume significant system resources. (Note, however, that resource consumption is much reduced in PMDF V5.0 as compared with previous versions of PMDF.) The maxprocchars keyword is used to specify the maximum length header that PMDF will process and rewrite. Messages with headers longer than this are still accepted and delivered; the only difference is that the long header lines are not rewritten in any way. A single integer argument is required. The default is to process headers of any length.

2.3.4.80 Mail delivery to over quota users (exquota, noexquota, holdexquota)

The exquota, noexquota, and holdexquota keywords control the handling of messages addressed to VMS MAIL mailbox users (OpenVMS), Berkeley mailbox users (UNIX), and PMDF popstore or PMDF MessageStore users (all platforms) who have exceeded their disk quotas.

exquota tells PMDF to ignore user quota limits and deliver messages even when users are over quota (except on the UNIX l channel, where exquota is equivalent to holdexquota; either keyword results in holding messages for over quota users). In particular, on OpenVMS for the l channel, PMDF uses the EXQUOTA privilege when exquota is used to perform the delivery to over quota users. noexquota tells PMDF to return messages addressed to over quota users to the message's sender. holdexquota tells PMDF to hold messages to over quota users; such messages will remain in the PMDF queue until they can either be delivered or they time out and are returned to their sender by the message return job. The default is exquota (which on UNIX for the l channel is equivalent to holdexquota). Use of the default is *strongly* recommended; bouncing mail on the basis of quota is usually not a good idea.

VMS

On OpenVMS, besides the l, msgstore, and popstore channels, these keywords technically also affect the other channels that deliver via VMS MAIL: the d channel and any mail_ channels. However, the keywords are typically not useful on d or mail_ channels since such channels typically connect to a remote transport agent of some kind and enabled privileges are not transferred.

2.3.4.81 Gateway daemons (daemon)

The interpretation and usage of the daemon keyword depends upon the type of channel to which it is applied.

DECUS UUCP channels

The daemon keyword is used on DECUS UUCP channels (vn_) to specify the name of the remote host to which the channel connects. This in turn makes it possible to have multiple channels that connect to the same remote system. If no daemon is specified, the remote host is derived from the channel name.

Local, DECnet MAIL-11, and MAIL channels

The daemon keyword is used on VMS MAIL channels (l, d, or mail_) to control certain aspects of address rewriting. See, for instance, Section 18.1 where special handling of DECnet mail and PSIMail addresses is discussed.

TCP/IP channels

Finally, the daemon keyword is also used on SMTP channels to control the choice of target host. Normally such channels connect to whatever host is listed in the envelope address of the message being processed. The daemon keyword is used to tell the channel to instead connect to a specific remote system, generally a firewall or mailhub system, regardless of the envelope address. The actual remote system name should appear directly after the daemon keyword, *e.g.*,

```
tcp_firewall smtp mx daemon firewall.example.com
TCP-DAEMON
```

If the argument after the daemon keyword is not a fully qualified domain name, the argument will be ignored and the channel will connect to the channel's official host. When specifying the firewall or gateway system name as the official host name, the argument given to the daemon keyword is typically specified as router, *e.g.*,

```
tcp_firewall smtp mx daemon router
firewall.example.com
TCP-DAEMON
```

2.3.4.82 Multiple gateways on a single channel (multigate, nomultigate)

The multigate keyword tells PMDF to route the message to the daemon mailbox specified by the daemon keyword (described in Section 2.3.4.81) on the system specified in the message's To: address. This differs from PMDF's behavior when the multigate keyword is not used, in which case PMDF routes the message to the official host associated with the channel, *not* the system specified in the message's To: address.

There are a variety of caveats associated with using the multigate keyword; some of its former uses are now obsolete. nomultigate is the default.

2.3.4.83 Grey Book address formatting (grey, nogrey)

The Grey Book protocol suite uses address formats that are similar to RFC 822, except that domains are specified in the opposite order; *e.g.,* user@relay.cs.net becomes user@net.cs.relay.

PMDF provides support for this format on a per-channel basis. If the keyword grey is specified in the channel block, then all addresses in both the header and the envelope of any message queued to the channel will be written in Grey Book format. This facility is disabled by default. (The nogrey keyword is the default, meaning that normal address format is used.)

Grey Book mail systems typically do not use RFC 822 source routes but use the RFC 733 percent-style routing addresses instead, so the `733` and `header_733` keywords are usually specified in addition to the `grey` keyword.

Note that the transformation is performed as messages enter and leave the channel; the rest of PMDF (in particular the PMDF configuration and alias files) always use the conventions of RFC 822.

Note: None of the channel programs provided in the standard PMDF distribution are designed to work with Grey Book addresses.

2.3.4.84 Message logging (`logging`)

PMDF provides facilities for logging each message as it is enqueued and dequeued. All log entries are made to the file `mail.log_current` in the PMDF log directory, (*i.e.*, `PMDF_LOG:mail.log_current` on OpenVMS or `/pmdf/log/mail.log_current` on UNIX. Logging is controlled on a per-channel basis. The `logging` keyword activates logging for a particular channel while the `nologging` keyword disables it. Logging is disabled on all channels by default.

The message return job, which runs every night around midnight, appends any existing `mail.log_yesterday` to the cumulative log file, `mail.log`, renames the current `mail.log_current` file to `mail.log_yesterday`, and then begins a new `mail.log_current` file.

The log file is written as normal ASCII text and the format is quite simple; see Section 31.1.2 for details.

If you want to have all of your channels log message activity to the logging file, then simply add a defaults channel block to the start of the channel block section of your PMDF configuration file. For instance,

```
defaults logging

l defragment charset7 us-ascii charset8 dec-mcs
example.com
```

The defaults channel would appear immediately after the first blank line in the PMDF configuration file. It is important that a blank line appear *before and after* the line "`defaults logging`". See Section 2.3.5 for a full description of the defaults channel.

Additional discussion of logging and examples of interpreting log file entries may be found in Section 31.1. That section also discusses managing the log files; in particular, note that PMDF itself never does anything to the cumulative `mail.log` file and it is up to each site to manage (*e.g.*, delete, truncate, backup, *etc.*) that log file however they choose.

2.3.4.85 Debugging channel master and slave programs (master_debug, nomaster_debug, slave_debug, noslave_debug)

Some channel programs include optional code to assist in debugging by producing additional diagnostic output. Two channel keywords are provided to enable generation of this debugging output on a per-channel basis. The keywords are master_debug, which enables debugging output in master programs, and slave_debug, which enables debugging output in slave programs. Both types of debugging output are disabled by default, corresponding to nomaster_debug and noslave_debug.

When activated, debugging output ends up in the log file associated with the channel program. The location of the log file may vary from program to program. Log files are usually kept in the PMDF log directory.[i] Master programs usually have log file names of the form *x*_master.log, where "*x*" is the name of the channel;[j] slave programs usually have log file names of the form *x*_slave.log. Also, some channel programs, notably PhoneNet channel programs, may produce additional log files with names of the forms:

err_*x*_master.log,	err_*x*_slave.log,
di_*x*_master.log,	di_*x*_slave.log, or
ph_*x*_master.log,	ph_*x*_slave.log.

Note that in the case of the l (lowercase "L") channel, master_debug enables debugging output when sending from the local channel (*e.g.,* from VMS MAIL), and slave_debug enables debugging output as messages are delivered to the local channel (*e.g.,* to VMS MAIL) (with output usually appearing in PMDF_LOG:l_master.log on OpenVMS or in pmdf/log/l_master.log on UNIX). On OpenVMS, these conventions also apply to the other channels that interact with VMS MAIL (d, d_, and mail_ channels). The thing to note is that this usage of the debug keywords is essentially backwards; other channels assign opposite meanings to the debug keywords. This usage is retained for historical and compatibility reasons.

On UNIX, when master_debug and slave_debug are enabled for the l channel, then users will get pmdf_sendmail.log-*uniqueid* files in their current directory (if they have write access to the directory; otherwise the debug output will go to stdout) containing PMDF debug information.

Not all PMDF channel programs have debugging support code.

[i] PMDF_LOG: on OpenVMS; /pmdf/log/ on UNIX.

[j] Note that the multithreaded TCP SMTP channel program will produce multiple tcp_*y*_master.log files per master channel program execution when master_debug is enabled. The first such file produced shows the channel's determination of how many outgoing threads to start up; an additional log file will be created for each individual outgoing thread.

2.3.4.86 Filter file location (filter, nofilter, channelfilter, nochannelfilter, destinationfilter, nodestinationfilter, sourcefilter, nosourcefilter, fileinto, nofileinto)

The filter keyword may be used on the l (lowercase "L"), msgstore, and popstore channels to specify the location of user filter files for that channel. It takes a required URL argument describing the filter file location. For details on the URL format and the general use of this keyword, see Section 16.2.1. nofilter is the default and means that user mailbox filters are not enabled for the channel.

The sourcefilter and destinationfilter keywords may be used on general PMDF channels to specify a channel-level filter to apply to incoming and outgoing messages, respectively. These keywords take a required URL argument describing the channel filter file location. For details on the URL format and the general use of these keywords, see Section 16.2.2. nosourcefilter and nodestinationfilter are the defaults and mean that no channel mailbox filter is enabled for either direction of the channel.

The obsolete channelfilter and nochannelfilter keywords are synonyms for destinationfilter and nodestinationfilter, respectively.

The fileinto keyword, currently supported only for msgstore channels when delivering to the PMDF MessageStore, specifies how to alter an address when a mailbox filter fileinto operator is applied. For msgstore channels, the usual usage is

```
fileinto $U+$S@$D
```

meaning that the folder name should be inserted as a subaddress into the original address, replacing any originally present subaddress.

2.3.4.87 Channel description field (description)

The description channel keyword provides a way to associate a descriptive term with a channel. This feature is intended for future management utility use.

2.3.4.88 Sensitivity checking (sensitivitynormal, sensitivitypersonal, sensitivityprivate, sensitivitycompanyconfidential)

The sensitivitynormal, sensitivitypersonal, sensitivityprivate, and sensitivitycompanyconfidential keywords set an upper limit on the sensitivity of messages that may be accepted by a channel. The default is sensitivitycompany-confidential; *i.e.*, messages of any sensitivity are allowed through. A message with no Sensitivity: header is considered to be of normal, i.e., lowest, sensitivity. Messages with a higher sensitivity than that specified by such a keyword will be rejected when enqueued to the channel with an error:

```
message too sensitive for one or more paths used
```

Note that PMDF does this sort of sensitivity checking at a per-message, not per-recipient, level: if a destination channel for one recipient fails the sensitivity check, then the message bounces for all recipients, not just for those recipients associated with the sensitive channel.

2.3.4.89 Access rights and privileges (`network`)

Unrecognized keywords are interpreted as rightslist identifiers (on OpenVMS) or as groups ids (on UNIX). On OpenVMS, a rightslist identifier as a channel keyword means that the user must be granted that rightslist identifier before they can queue messages to the channel; on UNIX, a group id as a channel keyword means that the user must be a member of that group before they can queue messages to the channel. See also the more flexible and general SEND_ACCESS mapping described in Section 16.1, or on OpenVMS systems only, the `network` channel keyword described below.

The PMDF TEST/REWRITE (OpenVMS) or `pmdf test -rewrite` (UNIX and NT) utility will tell you if you have any unrecognized keywords in your configuration file that don't match a known rightslist identifier (OpenVMS) or group id (UNIX).

VMS

The basic idea for using OpenVMS rightslist identifiers is as follows. Each channel specified in the configuration file can have one or more OpenVMS rightslist identifiers associated with it. These identifiers are specified as keywords on the same line as the channel name. If such an identifier is specified, PMDF checks to make sure that the identifier is held by the user. If it is not, the user cannot queue messages to that channel. The user must hold all the identifiers associated with a channel in order to be able to use that channel.

OpenVMS rightslist identifiers are created and managed with the AUTHORIZE utility and are the basis of OpenVMS system security. PMDF's use of identifiers follows OpenVMS guidelines. Rightslist identifiers used by PMDF should contain one or more dollar signs to prevent conflicts with future PMDF keyword definitions (PMDF keywords do not contain dollar signs). Consult HP's *Guide to OpenVMS System Management and Daily Operation* for additional information on rightslist identifiers.

On OpenVMS systems, if the `network` rightslist identifier is specified on a channel, it is specially interpreted by PMDF: PMDF will not allow users without network privileges (NETMBX) to queue messages to the channel.

2.3.4.90 Directory Channel Lookup Mode (`inline`, `noinline`)

Normally, look ups using the directory channel are queued to the directory channel itself and processed by the directory channel master program. When the channel keyword `inline` is specified on the directory channel, then look ups are done immediately, similar to the way look ups are done to the alias database. `noinline` is the default. These keywords only apply to directory channels. See Section 3.2.2 for more information.

2.3.4.91 Detecting Mail Loops (loopcheck)

When PMDF is sending mail, it can get into a loop sending mail back to itself if the destination domain has an MX record of 0.0.0.0 or 127.0.0.1. The `loopcheck` keyword causes PMDF to use the SMTP XLOOP line on the EHLO response so that PMDF can detect that it is sending mail back to itself, thus preventing a loop.

2.3.4.92 Accepting All Addresses (acceptalladdresses, acceptvalidaddresses)

Keyword `acceptvalidaddresses` is the default and corresponds to PMDF's standard behavior. If keyword `acceptalladdresses` is specified on a channel, then all recipient addresses are accepted during the SMTP dialogue. Any invalid addresses will have a Non-Delivery Notice sent later.

2.3.4.93 Relaxed Header Termination (relaxheadertermination, norelaxheadertermination)

Keyword `relaxheadertermination` is the default and corresponds to PMDF's standard behavior, which is to treat a line containing only spaces and tabs as meaning the same as a blank line, i.e. it terminates the header and the rest of the message is considered the body. If keyword `norelaxheadertermination` is specified on a channel, then a line within the header containing only spaces and tabs is treated as a continuation of the previous header line, and PMDF continues to process the next lines as part of the header.

2.3.4.94 Handle addresses from VMS MAIL (OpenVMS) (addlineaddrs, noaddlineaddrs)

By default, for messages sent from VMS MAIL, PMDF only adds addresses to its To: and Cc: headers that are processed by PMDF itself. Other addresses processed by other protocols (such as DECnet) are included only in the X-VMS-To: and X-VMS-Cc: headers. (This corresponds to `noaddlineaddrs`). Specify the `addlineaddrs` keyword on the `l` channel to tell PMDF to include all addresses on its To: and Cc: lines.

Make use of this functionality with caution. The address formats for other protocols may not map easily into the SMTP address format, rendering them invalid.

2.3.5 Using defaults and nodefaults channel blocks to simplify configurations

Many configurations involve repetition of various channel keywords on all or nearly all channels. Maintaining such a configuration is both tedious and error-prone. PMDF offers a simple way to change what defaults apply to various channel keywords. This mechanism can be used to greatly simplify some configurations.

If a line of the form:

```
defaults keyword1 keyword2 keyword3 ...
```

is inserted into the configuration, all channel blocks following this line will "inherit" the keywords specified on the line. The defaults line can be thought of as a special channel block that changes the keyword defaults without actually specifying a channel. The defaults line also does not require any additional lines of channel block information (if any are specified they will be ignored).

There is no limit on the number of defaults lines that can be specified — the effects of multiple defaults lines are cumulative with the most recently encountered (reading from top to bottom) line having precedence.

It may be useful to unconditionally eliminate the effects of any defaults lines starting at some point in the configuration file (at the start of a standalone section of channel blocks in an external file, for example). The nodefaults line is provided for this purpose. It takes the form:

```
nodefaults
```

and has the obvious effect — it nullifies all settings established by any previous defaults channel and returns the configuration to the state that would apply if no defaults had been specified.

Like regular channel blocks, a blank line must separate each defaults or nodefaults channel block from other channel blocks. The defaults and nodefaults channel blocks are the only channel blocks which may appear before the local channel in the configuration file. However, like any other channel block, they must appear after the last rewrite rule.

2.3.6 Available channels

Every PMDF channel has a unique name containing up to 32 characters. Only lowercase letters, numbers, underscores, and dollar signs should be used in channel names.

Certain channel names are reserved for particular uses. These reserved names are shown in Table 2–7.

Table 2–7 Reserved Channel Names

Name	Reserved For
`address`	Extract addressing information from the body of a message
`bitbucket`	Bit bucket channel (deletes all messages queued to it)
`circuitcheck`	Message circuit checking channel
`conversion`	Message body part conversion channel
`d`	The DECnet MAIL channel; used to deliver messages across DECnet via VMS MAIL
`data_to_bitmap`	Raw FAX data to bitmap channel

Table 2–7 (Cont.) Reserved Channel Names

Name	Reserved For
defragment	Message defragmentation channel
directory	Directory alias expansion channel
fax_to_data	Inbound FAX to raw data channel
filter_discard	Channel for discarding filter-discarded messages
g3_to_fax	Group 3 to FAX modem spooler
l	The local channel on OpenVMS and UNIX; used to deliver mail to users of the local system (and on OpenVMS, systems accessible via DECnet MAIL or PSIMail)
mailserv	Mail and list server channel
mime_to_x400	MIME to X.400 conversion channel
mint	MINT user agent from Wesleyan University
msgstore	PMDF Message Store delivery channel
p	Generic PhoneNet channel; used to communicate with a central PhoneNet host
pager	E-mail to pager channel
pipe	Pipe channel
popstore	PMDF popstore delivery channel; a msgstore channel can be—and typically is – used instead
ps_to_g3	PostScript to Group 3 FAX interpreter
printer	e-mail to spooled printer
process	Processing channel
text_to_ps	Text to PostScript converter
reprocess	Reprocessing channel
subject	Channel to extract addresses from Subject: lines
x400_to_mime	X.400 to MIME conversion channel
x400_local	X.400 transport channel

Moreover, certain families of channel names are assumed to be of particular types. Special channel programs will be invoked to service channels whose names begin with the prefixes listed in Table 2–8. (Note that some of these reserved names correspond to third party channels or obsolete channels, rather than to channels currently or by default available with PMDF.)

Table 2–8 Reserved Channel Name Prefixes

Prefix	Type of Channel
aoce_	Apple AOCE channels
address_	Addressing channels
anje_	ANJE (BITNET)
bit_	Jnet (BITNET)
bsin_	BSMTP inbound channels
bsout_	BSMTP outbound channels
bull_	BULLETIN channels
cn_	Internal usage by the national Australian network
ctcp_	Carnegie Mellon University TCP/IP channels; obsolete
data_to_bitmap_	Data to bitmap channels
d_	MAIL-11 over DECnet
directory_	Directory alias expansion channels
dn_	PhoneNet over DECnet
dsmtp_	SMTP over DECnet

Table 2–8 (Cont.) Reserved Channel Name Prefixes

Prefix	Type of Channel
era_	ERA channels
etcp_	Excelan TCP/IP channels; obsolete
ftcp_	Network Research Corporation FUSION TCP/IP channels; obsolete
ker_	Kermit protocol
mail_	General VMS MAIL delivery
msgstore_	PMDF MessageStore channel
mtcp_	Process Software MultiNet (formerly Cisco MultiNet, formerly TGV MultiNet) TCP/IP channels; obsolete
notes_	DEC NOTES channels
osfl_	UNIX local channels
p_	PhoneNet channels
pager_	Pager channels
pipe_	Pipe channels
printer_	e-mail to spooled printer channels
process_	Processing channels
ptcp_	Process Software TCPware
px25_	PhoneNet over X.25; obsolete
qm_	QuickMail channels
reprocess_	Reprocessing channels
snads_	SNADS channels
sync_db_	Database synchronization channels
sync_dirbot_	Directory synchronization robot (DIRBOT) channels
sync_ldap_	LDAP directory synchronization channels
sync_ldif_	LDIF directory agent channels
sync_ln_	Lotus Notes directory agent channels
tcp_	Multithreaded TCP/IP SMTP channels
test_	Test channels
text_to_ps_	Text to PostScript channels
tcp_	Multithreaded TCP/IP channels
utcp_	ULTRIX (UCX) Connection TCP/IP channels; obsolete
uucp_	UUCP channel (UNIX, or DEC/Shell UUCP)
vn_	UUCP channel (DECUS UUCP)
wtcp_	Wollongong TCP/IP (WIN/TCP) channels; obsolete

Note that no channel programs for cn or ker channels are included in the standard PMDF distribution; these two channels are, respectively, provided by the administrators of the Australian national network and Fel Computing.

These reserved channel names and prefixes are used internally by PMDF, especially by the central master program dispatcher, (as for instance `PMDF_COM:master.com` on OpenVMS or the PMDF Job Controller on UNIX and NT). Using names in a conflicting manner can lead to serious problems. System managers are encouraged to use these channels for the stated purposes and in general to pick channel names of their own that do not conflict with these usage conventions.

2.3.7 Header option files

Some special option files may be associated with a channel that describe how to trim the headers on messages queued to that channel. This facility is completely general and may be applied to any channel; it is controlled by the `headertrim`, `noheadertrim`, `headerread`, and `noheaderread` channel keywords.

Various PMDF channels have their own channel-level option files as well. Header option files have a different format than other PMDF option files and thus a header option file is always a separate file.

2.3.7.1 Header option file location

For destination channel based header trimming to be applied upon message enqueue after normal header processing, PMDF looks in the table directory, `PMDF_TABLE:` on OpenVMS or `/pmdf/table/` on UNIX, for header options files with names of the form *`channel`*`_headers.opt`, where *`channel`* is the name of the channel with which the header option file is associated. The `headertrim` keyword must be specified on the channel to enable the use of such a header option file.

For source channel based header trimming to be applied upon message enqueue before normal header processing, PMDF looks in the table directory, `PMDF_TABLE:` on OpenVMS or `/pmdf/table/` on UNIX, for header options files with names of the form *`channel`*`_read_headers.opt`, where *`channel`* is the name of the channel with which the header option file is associated. The `headerread` keyword must be specified on the channel to enable the use of such a header option file.

Header option files should be world readable.

2.3.7.2 Header option file format

Simply put, the contents of a header option file are formatted as a set of message header lines. Note, however, that the bodies of the header lines do not conform to RFC 822.

The general structure of a line from a header options file is then:

`Header-name``: OPTION=VALUE, OPTION=VALUE, OPTION=VALUE, ...`

where *`Header-name`* is the name of a header line that PMDF recognizes (any of the header lines described in this manual may be specified, plus any of the header lines standardized in RFC 822, RFC 987, RFC 1049, RFC 1421, RFC 1422, RFC 1423, RFC 1424, RFC 2156, and RFC 2045).

Header lines not recognized by PMDF are controlled by the special header line name Other:. A set of options to be applied to all header lines not named in the header option file can also be given on a special Defaults: line. Use of Defaults: guards against the inevitable expansion of PMDF's known header line table in future releases.

Various options may then be specified to control the retention of the corresponding header lines. The available options are:

ADD (quoted string)
The ADD option creates a completely new header line of the given type. The new header line contains the specified string. The header line created by ADD will appear after any existing header lines of the same type. The ADD option cannot be used in conjunction with the Defaults: header line type; it will be ignored if it is specified as part of an Other: option list.

CUTLINES (integer)
This option controls the maximum number of lines all header lines of a given type may occupy. It complements the MAXIMUM option in that it pays no attention to how many header lines are involved, only to how many lines of text they collectively occupy. As with the MAXIMUM option, headers are trimmed from the bottom to meet the specified requirement.

CUTLINES is similar to the MAXLINES option, but it will cut off a header instance in the middle. For that reason, MAXLINES is recommended over CUTLINES. CUTLINES is really only useful for the PMDF MAIL header trimming option file because it controls the display to the screen instead of actual headers contained in mail messages.

Note: CUTLINES is **not** recommended for use in channel header option files. It is recommended for use **only** in the PMDF MAIL header option file.

EMPHASIS (integer)
This option adds emphasis to the display of the header label. It is only useful for use by PMDF MAIL, when displaying headers on the screen. It is a bit mask with the bits defined as follows: 1=bold, 2=underline, 4=reverse.

Note: EMPHASIS is **not** recommended for use in channel header option files. It is recommended for use **only** in the PMDF MAIL header option file.

FILL (quoted string)
The FILL option creates a completely new header line of the given type if and only if there are no existing header lines of the same type. The new header line contains the specified string. The FILL option cannot be used in conjunction with the header line type; it will be ignored if it is specified as part of an Other: option list.

GROUP (integer 0 or 1)
This option controls grouping of header lines of the same type at a particular precedence level. A GROUP value of 0 is the default, and indicates that all header lines of a particular type should appear together. A value of 1 indicates that only one header line of the respective type should be output and the scan over all header lines at the associated level should resume, leaving any header lines of the same type unprocessed. Once the scan is complete it is then repeated in order to pick up any remaining header lines. This header option is primarily intended to accomodate Privacy Enhanced Mail (PEM) header processing.

LINELENGTH (integer)
This option controls the length at which to fold headers. See also the discussion of the `headerlinelength` channel keyword in Section 2.3.4.75.

MAXCHARS (integer)
This option controls the maximum number of characters which may appear in a single header line of the specified type. Any header line exceeding that length is truncated to a length of MAXCHARS. This option pays no attention to the syntax of the header line and should never be applied to header lines containing addresses and other sorts of structured information. The length of structured header lines should be controlled with the `maxheaderchars` and `maxheaderaddrs` channel keywords.

MAXIMUM (integer)
This option controls the maximum number of header lines of this type that may appear. This has no effect on the number of lines, after wrapping, each individual header line might consume. A value of -1 is interpreted as a request to suppress this header line type completely.

MAXLINES (integer)
This option controls the maximum number of lines all header lines of a given type may occupy. It complements the MAXIMUM option in that it pays no attention to how many header lines are involved, only to how many lines of text they collectively occupy. As with the MAXIMUM option, headers are trimmed from the bottom to meet the specified requirement.

MAXLINES is similar to the CUTLINES option, but MAXLINES will not stop in the middle of an instance of a header. It will output all lines of that instance of the header, even if it goes a few lines beyond MAXLINES lines. This option is recommended for use over CUTLINES for that reason.

PRECEDENCE (integer)
This option controls the order in which header lines are output. All header lines have a default precedence of zero. The smaller the value, the higher the precedence. Thus, positive PRECEDENCE values will push header lines towards the bottom of the header while negative values will push them towards the top. Equal precedence ties are broken using PMDF's internal rules for header line output ordering.

RELABEL (header name)
This option changes a header line to another header line; that is, the name of the header is changed, but the value remains the same. For instance,

```
X-MSMail-Priority: RELABEL="Priority"
X-Priority: RELABEL="Importance"
```

2.4 Some example configuration files

This section contains four sample configurations. The first example, Example 2–2 illustrates the use of rewrite rules. The second example, Example 2–3, illustrates a configuration which may be used on a satellite cluster or node which routes all non-local mail to a hub. The third example, Example 2–4, shows a sample configuration for a site connected to the Internet. And the final example, Example 2–5, demonstrates the handling of a local network in which only one of the machines is running PMDF and the other machines are networked via DECnet.

2.4.1 A simple configuration file

The following example configuration file for OpenVMS or UNIX (since an l channel is used) shows how rewrite rules are used to route messages to the proper channel. No domain names are used in order to keep things as simple as possible.

Example 2–2 A Simple Configuration File

```
! test.cnf - An example configuration file for PMDF. ❶
!
! This is only an example of a configuration file. It serves
! no useful purpose and should not be used in a real system.
!
a     $U@a     ❷
b     $U@b
c     $U%c@b
d     $U%d@a
                ❸
l               ❹
local-host

a_channel 822 ❺
a
e

b_channel 733 network
b
f
```

The key items in the configuration file shown in Example 2–2 are

❶ Exclamation points, !, are used to introduce comment lines. The exclamation point *must* appear in the first column. An exclamation point appearing anywhere else is interpreted as a literal exclamation point.

❷ The rewrite rules appear in the first half of the configuration file. Absolutely no blank lines should appear amongst the lines of rewrite rules. Lines with comments (beginning with an exclamation point in the first column) are, however, permitted.

❸ The first blank line to appear in the file signifies the end of the rewrite rules section and the start of the channel blocks.

❹ The first channel block to appear is always the local channel. On OpenVMS and UNIX platforms, this is the "l" channel (lowercase letter L). Blank lines then separate each channel block from one another. Exception: a defaults channel may appear before the local channel.

❺ A channel named a_channel. Note the use of a channel keyword (822) with this channel.

The routing and queuing of messages by the configuration seen in Example 2–2 is shown in Table 2–9 below.

Table 2–9 Message Routing and Queuing Generated by Example 2–2

Address	Routed to	Queued to channel
u@a	a	a_channel
u@b	b	b_channel
u@c	b	b_channel
u@d	a	a_channel
u@e	e	a_channel
u@f	f	b_channel

2.4.2 Routing non-local mail to a central mail hub

Sometimes it is convenient to configure PMDF to route mail not for the local host, or a group of local machines, to a central machine and leave it up to that machine to deal with the mail, perhaps relaying it to the outside world or other local machines, or perhaps even gatewaying it into other mail systems. The following example configuration, Example 2–3, illustrates doing just this. The local host is HOSTA.EXAMPLE.COM and two other local machines, HOSTB.EXAMPLE.COM and HOSTC.EXAMPLE.COM, are recognized. Mail for either of those two machines is sent via a tcp_local channel (SMTP over TCP/IP) to those hosts. All other mail not for HOSTA, HOSTB, or HOSTC is sent via another SMTP over TCP/IP channel, named tcp_gateway, to the host MAILHUB.EXAMPLE.COM. A "match-all" rule is used to direct all mail not for HOSTA, HOSTB, or HOSTC to that channel. (The match-all rule is described in Section 2.2.4.3.) The `daemon` keyword is used with the tcp_gateway channel; the usage of this keyword with SMTP over TCP/IP channels is discussed in Section 2.3.4.81. It tells the channel to route messages queued to it through the host MAILHUB.EXAMPLE.COM.

Example 2–3 Routing Messages to a Central Machine

```
!
! Rewrite rules for the local host/cluster
!
HOSTA                               $U@HOSTA.EXAMPLE.COM
HOSTA.EXAMPLE.COM                   $U@HOSTA.EXAMPLE.COM
!
! Rewrite rules for some internal systems
!
HOSTB.EXAMPLE.COM                   $U%HOSTB.EXAMPLE.COM@TCP-DAEMON
HOSTC.EXAMPLE.COM                   $U%HOSTC.EXAMPLE.COM@TCP-DAEMON
!
! Use a match all rule to route everything
! else to the MAILHUB.EXAMPLE.COM
!
.                                  $U%$H@MAILHUB.EXAMPLE.COM$A
```

Example 2–3 Cont'd on next page

Example 2–3 (Cont.) Routing Messages to a Central Machine

```
l
HOSTA.EXAMPLE.COM

tcp_local smtp single_sys mx
TCP-DAEMON

tcp_gateway smtp mx daemon router
MAILHUB.EXAMPLE.COM
```

2.4.3 Basic configuration for a system on the Internet

The configuration file shown in Example 2–4 is a minimal version of the sort of configuration typical for a system that communicates directly with the Internet via TCP/IP. The system's name in this example is assumed to be sample.com.

Example 2–4 Sample Configuration File

```
! pmdf.cnf - PMDF configuration file for sample.com.
!
! Rewrite rules for the local host/cluster
!
sample                          $U@sample.com
sample.com                      $U@sample.com
!
! Rewrite rules for the Internet
!
!    Ascension Island
.AC                                         $U%$H$D@TCP-DAEMON
. [text
.    removed for
.             brevity]
! Zimbabwe
.ZW                             $U%$H$D@TCP-DAEMON
!
! BITNET (not properly an Internet domain)
!
.BITNET                         $U@$H$D@interbit.cren.net@TCP-DAEMON
!
! Rewrite rules for TCP/IP domain literals
!
[1.2.3.4]                       $U@sample.com
[]                              $U%[$L]@TCP-DAEMON
```

```
l nox_env_to
sample.com

tcp_local single_sys smtp mx
TCP-DAEMON
```

This configuration file is quite simple. Messages to the local system (whose name is "sample.com") are sent to the local channel. Any other message whose address contains a recognizable top-level domain specification is routed through the SMTP over TCP/IP channel, tcp_local, to the Internet. Any other address is treated as being illegal.

2.4.4 Handling systems on a local DECnet (OpenVMS)

The following example configuration file shows how to set up a configuration that provides access to a number of OpenVMS systems accessible via DECnet but not themselves running PMDF. Although these remote machines are not running PMDF they can, nevertheless, participate fully in the network. The only disadvantage to not running PMDF on all the remote systems is that the remote systems cannot send mail to the gateway unless the gateway system is up. The converse is not true; messages will be queued on the gateway until they can be delivered to the remote systems.

The domain name of the gateway system is MILAN.EXAMPLE.COM. Its DECnet name is MILAN. The names of the remote systems are TUSCANY.EXAMPLE.COM (DECnet name TUSCAN), CMCVAX.EXAMPLE.COM (CMCVAX), ECHMC.EXAMPLE.COM (ECHMC), MEDCHEM.EXAMPLE.COM (MEDCHM). MEDCHEM.EXAMPLE.COM is also known under the alias MEDCHM.EXAMPLE.COM. Shortform aliases equivalent to the DECnet node names are also provided. Note that the DECnet node names are not necessarily the same as the first part of the domain names.

Simple rewrite rules are used to map the names of the systems and any aliases or shortform names into the proper canonical domain names. This insures that the proper domain-style addresses appear in all message headers.

Channel table rewriting is then used to map the domain names back onto the proper DECnet node names. Channel rewriting is only applied to envelope To: addresses and hence only affects the addresses that need to be converted to DECnet node name format; all other addresses are left unchanged.

Without further ado, then, the requisite configuration is exhibited in Example 2–5 below.

Example 2–5 Configuring a Gateway for a DECnet Network

```
MILAN.EXAMPLE.COM                $U@MILAN.EXAMPLE.COM
MILAN                            $U@MILAN.EXAMPLE.COM
```

Example 2–5 Cont'd on next page

Example 2–5 (Cont.) Configuring a Gateway for a DECnet Network

```
!
TUSCANY.EXAMPLE.COM                $U@TUSCANY.EXAMPLE.COM
TUSCANY                            $U@TUSCANY.EXAMPLE.COM
TUSCAN                             $U@TUSCANY.EXAMPLE.COM
CMCVAX.EXAMPLE.COM                 $U%CMCVAX.EXAMPLE.COM@DECNET-MAIL
CMCVAX                             $U%CMCVAX.EXAMPLE.COM@DECNET-MAIL
ECHMC.EXAMPLE.COM                  $U%ECHMC.EXAMPLE.COM@DECNET-MAIL
ECHMC                              $U%ECHMC.EXAMPLE.COM@DECNET-MAIL
MEDCHM.EXAMPLE.COM                 $U@MEDCHEM.EXAMPLE.COM
MEDCHM                             $U@MEDCHEM.EXAMPLE.COM
MEDCHEM.EXAMPLE.COM                $U@MEDCHEM.EXAMPLE.COM
MEDCHEM                            $U@MEDCHEM.EXAMPLE.COM

l
MILAN.EXAMPLE.COM

d 733
DECNET-MAIL
TUSCANY.EXAMPLE.COM  TUSCAN
MEDCHEM.EXAMPLE.COM  MEDCHM
```

3 Aliases, Forwarding, and Centralized Naming

PMDF provides a facility to support mailbox names associated with the local system that do not necessarily correspond to actual users. Such "aliases" are useful for constructing mailing lists, forwarding mail, and synonyms for usernames. A second set of related facilities provide support for "centralized naming" whereby you establish, for instance, mail addresses of the form *first.last*`@example.com` for all of your users. There are several advantages to such centralized naming systems: the addresses are simple, they provide added security in that they make no reference to internal account or system names, and, because they lack reference to account and system names, are more stable.

The concept of aliases, mailing lists, and mail forwarding are very closely related in PMDF as they are all effected through the use of PMDF's alias facilities, described below. Perhaps less obvious, is the relationship between mail forwarding and centralized naming schemes. To support centralized naming, a mailer must not only be able to convert internal addresses such as `jd001@vax1.example.com` into `John.Doe@example.com` in all outbound mail, but also be able to recognize incoming mail for `John.Doe@example.com` and forward it to `jd001@vax1.example.com`. Hence the relationship between centralized naming and mail forwarding and, in turn, aliases.

Finally, as will be pointed out in Sections 3.5 and 3.6, there are several different ways to effect forwarding and centralized naming. The different approaches vary in efficiency and which approach you can use will be largely dictated by the regularity of the mapping between internal and centralized addresses: the more susceptible to pattern matching a mapping is, the more efficiently it may be implemented.

3.1 Aliases and Forwarding

Each time an address that matches the `local` channel[1] or any channel marked with the `aliaslocal` keyword is encountered by PMDF's message submission logic, the mailbox (*e.g.,* username) specified in the address is compared against each entry in the alias database or alias file. If a match occurs the alias address is replaced by the translation value or values specified by the alias. An alias can translate into any combination and number of additional aliases or real addresses. The real addresses need not themselves be associated with the local channel and thus aliases can be used to forward mail to remote systems. If the translation value of an alias is a file name preceded by a <, then the contents of that file are used as a mailing list (*e.g.,* distribution list) and the message is sent to each recipient listed in the file; if the translation value of an alias is an LDAP URL preceded by a < that returns one or multiple addresses, then the message is sent to each address returned. This process is occasionally referred to as "mail exploding". See Section 4.1 for directions on how to set up a mailing list.

[1] The local channel is the `l` (lowercase L) channel on OpenVMS or UNIX, or the `msgstore` channel on NT.

Aliases only apply to addresses mapped to the local channel[1] or to channels marked with the aliaslocal keyword; furthermore, note that since the only addresses truly considered to match a channel are Envelope To: addresses, aliases can only apply to Envelope To: addresses. PMDF performs alias translation and expansion only after address parsing is completed. The translation values produced by an alias are treated as completely new addresses and are reprocessed from scratch.[2]

Aliases as well as mailing lists, a special case of an alias, may be tested with the command PMDF TEST/REWRITE/CHECK_EXPANSIONS (OpenVMS) or pmdf test -rewrite -check_expansions (UNIX or NT). See Chapter 29 and Chapter 30 for details. In regards to mailing lists, see the final part of Section 4.1.3.

3.1.1 The Alias File

Aliases are kept in a central file, usually aliases in the PMDF table directory.[3] Each time a PMDF program begins running, this file is read and loaded into an internal hash table. This overhead may be avoided by compiling your PMDF configuration in which case the contents of the alias file will be incorporated into the compiled configuration. The disadvantage to this, however, is that it means that the configuration must be recompiled and reinstalled whenever a change is made to the alias file or an include file used by the alias file. See Section 8.1 for details on compiling your configuration.

The alias file and any files it references should be world readable. Failure to allow world read access will lead to erratic behavior.

3.1.1.1 Format

The alias file format is as follows:

```
alias1: a1,a2,...,am
alias2: b1,b2,...,bm
   .              .
   .              .
   .              .
aliasn: n1,n2,...,nm
   .              .
   .              .
   .              .
```

where *aliasn* is translated into the addresses *n1*, *n2*, *n3*, ..., *nm*. The aliases *alias1*, *alias2*, ..., *aliasn* are limited to 60 characters each. Each address *a1*, *a2*, *etc.*, may contain up to 252 characters. There is no limit to the number of addresses that can be

[2] Sometimes it is desirable to have more than one set of aliases associated with the local channel or host. This situation is addressed by the directory channel discussed in Section 3.2.

[3] On OpenVMS systems, the PMDF alias file is pointed at by the PMDF_ALIAS_FILE logical; by default, it is the file PMDF_TABLE:aliases.. On UNIX systems, it is the file specified with the PMDF_ALIAS_FILE option in the PMDF tailor file; by default, the file /pmdf/table/aliases. On NT systems, it is the file specified with the PMDF_ALIAS_FILE NT Registry entry, usually pointing to a file such as C:\pmdf\table\aliases.

specified for an alias (that is, appear in a single list on the right hand side of an alias definition), although excessive numbers of addresses may eat up excessive amounts of memory. A physical line of the alias file may contain at most 1024 characters. To specify a list of addresses containing more than that number of characters, the line must be continued onto multiple physical lines. Long lines may be continued by ending them with a backslash, `\`. A backslash must follow a comma. There can be no white space preceding the colon separating the alias name from its translation value.

Alternatively, rather than having an address or comma separated list of addresses as the translation of an alias, an alias may translate to a mailing list reference as discussed in Section 3.1.1.3 below, or to an LDAP URL reference as discussed in Section 3.1.1.4 below.

Example 3–1, Example 3–2, and Example 3–3 show typical, minimal alias files on OpenVMS, UNIX, and NT, respectively.

An `alias` should normally be a valid RFC 822[4] local-part; however, the `ALIAS_DOMAINS` `PMDF` option (see Section 7.3.1) controls the format of aliases and use of a non-default value for `ALIAS_DOMAINS` can specify that aliases consist of the entire address, including the domain name, rather than just the local-part. In particular, aliases must follow RFC 822 syntax rules for local-parts (or addresses, when `ALIAS_DOMAINS` has selected use of addresses); this means that for proper functioning, with the exception of periods which are specifically allowed in local-parts without quoting, the presence of any other RFC 822 "specials" character or a space in an alias will require that the alias be enclosed in double quotes, *e.g.*,

```
"John Doe": doe@example.com
john.doe: doe@example.com
```

OpenVMS postmasters in particular should note also that RFC 822 addresses do not contain VMS MAIL's `IN%` wrapper; nor are DECnet style addresses (*e.g.*, *`NODE`*`::`*`USER`*) valid RFC 822 addresses.

Comment lines are allowed in the alias file. A comment line is any line that begins with an exclamation point, `!`, in column one.

Duplicate aliases (identical left hand sides) are not allowed in the alias file.

Note that certain sorts of errors in the format of aliases will not result in an immediate error message, but rather mail to the bad addresses will just be silently dropped; use `PMDF TEST/REWRITE` (OpenVMS) or `pmdf test -rewrite` (UNIX or NT) to check aliases, and see Section 3.1.5 for further general information on alias files.

[4] A copy of RFC 822, a basic reference for any e-mail administrator, is shipped with the PMDF distribution; it should be present as `PMDF_DOC:[rfc]rfc822.txt` on OpenVMS, or as `/pmdf/doc/rfc/rfc822.txt` on UNIX, or as `C:\pmdf\doc\rfc\rfc822.txt` on NT, unless your site chose not to install it.

3.1.1.2 Including Other Files in the Alias File

Other files can be included in the primary alias file. A line of the form

```
<file-spec
```

directs PMDF to read the file *file-spec*. The file specification must be a complete file path specification and the file must have the same protections as the primary alias file; *i.e.,* it must be world readable.

The contents of the included file are inserted into the alias file at its point of reference. The same effect can be achieved by replacing the reference to the included file with the file's actual contents. The format of include files is identical to that of the primary alias file itself. Indeed, include files may themselves include other files. Up to three levels of include file nesting are allowed.

If a compiled configuration is being used, then the configuration must be recompiled and reinstalled before changes to any included file (or the primary alias file itself) will take effect. Note that this is not the case for mailing list files described in the next section.

3.1.1.3 Mailing Lists

A mailing list address is a special address created through the `alias file` or `alias` database. A mailing list address *alias* with associated mailing list file *file-spec* or `LDAP URL` *ldap-url* is specified in the `alias` file with an entry of, respectively, the general form

```
alias: <file-spec, optional-parameters
```

or

```
alias: <ldap-url, optional-parameters
```

Similar definitions may also be made in the alias database, (though of course omitting the colon, as just white space separates the alias from its definition in the alias database).

Mailing lists have many options associated with them; for a full discussion of mailing list aliases, see Chapter 4.

3.1.1.4 LDAP URLs as Alias Values

An alias value (that is, the right hand side of an alias definition) may be specified either as an address directly, *e.g.*, *user@domain*, or indirectly referencing an LDAP URL—specifically, an LDAP search URL—that returns one or more addresses. The format is

```
alias: <"ldap-url"
```

Note: The LDAP URL must be specified in double-quotes if it contains any commas.

Note that this is just a special case of use of an `LDAP URL` for a mailing list definition, as mentioned in Section 3.1.1.3: the `LDAP` query URL may be such as to return only one

address rather than multiple addresses, and all of the optional mailing list parameters are omitted.

Standard LDAP URLs are used, with the host and port omitted; the host and port are instead specified with LDAP_HOST and LDAP_PORT PMDF options (see Section 7.3.2 for further discussion of these options). That is, the LDAP URL should be specified as

```
ldap:///dn[?attributes[?scope?filter]]
```

where the square bracket characters *[* and *]* shown above indicate optional portions of the URL. The *dn* is required and is a distinguished name specifying the search base. The optional *attributes*, *scope*, and *filter* portions of the URL further refine what information to return. For an alias, the desired *attributes* to specify returning would typically be the mail attribute (or some similar attribute). The *scope* may be any of base (the default), one, or sub. And the desired *filter* might be to request the return of any object that has the "objectclass=person" and "cn=John Smith" attribute-value pairs.

For instance, at a site example.com with an LDAP server running on port 389 of the system ldap.example.com, the PMDF option file might have the lines

```
LDAP_HOST=ldap.example.com
LDAP_PORT=389
```

set, and an alias file line might appear as:

```
John.Smith:  <"ldap:///o=example.com?mail?sub?(objectClass=person,cn=John Smith)"
```

Note that port number 389 is the default. Also note that LDAP URL is specified in double-quotes since it contains commas.

Substitution sequences, as shown in Table 3–1, are available for use in constructing the LDAP URL.

Table 3–1 LDAP URL Substitution Sequences

Substitution sequence	Description
$$	Literal $ character
$~*account*	Home directory of user *account*
$A	Address
$D	Domain name
$H	Host name (first portion of fully qualified domain name)
$L	Username minus any special leading characters such as ~ or _
$S	Subaddress
$U	Username

3.1.1.5 Standard Aliases

Certain aliases should be provided in every alias file. If no account with the name postmaster exists on the system, an alias for postmaster should be provided that translates into the username of the person responsible for maintaining PMDF (often, but not always, the `SYSTEM` account on OpenVMS systems or the root account on UNIX systems). It is also a good idea to provide an alias for postmast, since some mail systems cannot handle mailbox names with more than eight characters. In addition, on OpenVMS systems, an alias for root should also be provided since many UNIX systems send mail to root when attempting to contact the local system manager. A minimal alias file for an OpenVMS system would then be as shown in Example 3–1.

Example 3–1 A Minimal Alias File, `aliases.`, on OpenVMS

```
postmast: postmaster
postmaster: system
root: system
```

A minimal alias file for a UNIX system would be as shown in Example 3–2.

Example 3–2 A Minimal Alias File, `aliases`, on UNIX

```
postmast: postmaster
postmaster: root
```

A minimal alias file for an NT system would be as shown in Example 3–3.

Example 3–3 A Minimal Alias File, `aliases`, on NT

```
postmast: postmaster
postmaster: Administrator
```

3.1.1.6 Subaddresses in Aliases

As background on the purpose of subaddresses, the L, popstore, and msgstore channels interpret a + character in an address specially: in an address of the form *name+subaddress@localhost* or *name+subaddress@popstoredomain* PMDF considers the portion of the mailbox after the plus character a *subaddress*. The L and popstore channels treat a subaddress as additional cosmetic information and, assuming no aliases or other address transformations apply, actually delivers to the account *name* without regard to the subaddress. The msgstore channels treat a subaddress as specifying a folder to which to deliver; that is, msgstore channels deliver to the *name* account's *subaddress* folder.

When looking up an alias, the use of subaddresses introduces an extra factor. The PMDF local channel, that is, the L channel on OpenVMS or UNIX or the first msgstore channel on NT, or any channel marked with the `aliaslocal` keyword, will try looking up aliases.

Subaddresses in aliases are handled as follows. By default, (that is, with the `subaddressrelaxed` keyword explicitly or implicitly on the channel doing the alias lookup), PMDF first checks for an alias entry including the subaddress; if no such entry is found, PMDF next checks for an entry with an asterisk, `*`, in place of the subaddress. Finally, if there is no prior match, PMDF checks for an entry without any subaddress. For instance, alias entries

```
adam+privileged:    system
adam:               bob+*
carl+special:       system
carl+*:             david+*
carl:               eric
```

cause PMDF to translate `adam+privileged` to `system` and `adam` to `bob`, while `adam+talklist`, `adam+general`, *etc.*, will be translated to bob+talklist, bob+general, *etc.* carl+special will be translated to `system` and `carl` to `eric`, while `carl+talklist`, `carl+general`, *etc.*, will be translated to `david+talklist`, `david+general`, *etc.*

This handling of subaddresses during alias lookups is configurable; see Section 2.3.4.71.

3.1.1.7 Alias List Recursion

Aliases may reference other aliases, both in the alias database as well as in the alias file. PMDF limits such references to a maximum of ten levels to avoid possible infinite recursion loops.

If an alias references itself, either directly or indirectly, an alias loop results. The loop eventually terminates due to the level restriction, but the termination conditions may not produce consistent results in all cases.

The special case of an alias directly referencing itself is allowed and specially handled. For example, the alias file definition

alias-name: *alias-name*, *other-address-1*, *other-address-2*, ...

will expand *alias-name* into itself plus *other-address-1*, *other-address-2*, and so on. *alias-name* may in turn get expanded in some other way (the system or personal alias database) but it will not be expanded further by the alias file.

3.1.2 The Alias Database

PMDF always reads in all the aliases from the `alias` file and stores them internally in a hash table. This scheme is adequate for most applications where fewer than 500 or so aliases are needed.

However, some systems may want to establish aliases for a majority of their users. For example, a user `smith` whose real name is Cathy Smith, might want to have `Cathy_Smith`, `Cathy.Smith`, and `Smith` as aliases for her account. Setting up such aliases for each member of a large user population may lead to an excessively large `alias` file that consumes far too much memory.

PMDF solves this problem by providing an optional facility for storing large numbers of aliases in an ancillary indexed data file. Whenever the alias file is used PMDF also checks for the existence of the PMDF `alias` database.[5] If the `alias` database exists, it is opened and consulted once for each address on the local channel. The alias database must be world readable.

By default, the mere presence of the alias database is enough to activate this database facility in PMDF; it is not necessary to rebuild or reconfigure PMDF to include it. Use of the `alias` database can be disabled with the PMDF option `USE_ALIAS_DATABASE`. This option can also be used to tell PMDF that the alias database is required, and if it isn't there, to return a temporary error (by specifying USE_ALIAS_DATABASE=2).

3.1.2.1 Using Both the Alias File and the Alias Database

The `alias` database is a *supplement* to the alias file; it is *not* a *replacement* for the `alias` file. If the alias database exists, PMDF uses *both* the `alias` file and the `alias` database.

The alias database is consulted once each time the regular alias file is consulted. However, the alias database is checked *before* the regular alias file is used. In effect, the database acts as a sort of address rewriter that is invoked prior to using the regular alias file. Although duplicate entries are allowed in the database, it is undefined as to which of the duplicate entries will be returned when the database is accessed. Database entries are case insensitive.

The fact that limited recursion is allowed in the alias file makes the complete translation mechanism rather complex. For example, suppose that the alias file contains the entries,

```
A: C,J
B: D,K
D: G,H
E: I
```

and the `alias` database contains the entries,

```
D: E
C: B
F: D
```

Now suppose the address `A@`*`local-host`* was presented to PMDF. First A would be looked up in the database — not found. Then A would be translated into C and J by the alias file. C would in turn be translated into B by the database while J would remain unchanged. B would then be translated into D and K by the alias file. D would then be translated into E by the database while K would remain unchanged. Finally, E would be translated into I by the alias file, and since I does not appear in the database the process would terminate. The final result is that A translates into the list I, J, K.

[5] On OpenVMS systems, the logical `PMDF_ALIAS_DATABASE` points to the `alias` database, which by default is the file `PMDF_TABLE:aliases.dat`. On UNIX systems, the `PMDF_ALIAS_DATABASE` option in the PMDF tailor file points to the the `alias` database; by default, the file `/pmdf/table/aliasesdb.*`. On NT systems, the `PMDF_ALIAS_DATABASE` NT Registry entry points to the alias database, usually the file `C:\pmdf\table\aliasesdb.*`.

The easiest way to look at the translation process is to simply follow it step-by-step as illustrated below.

Initial look up	Data base	File	Data base	File	Data base	File	Data base	Result
A	A	C	B	D	E	I	I	I
				K	K	.	.	K
		J	J	.	.	.	.	J
B	B	D	E	I	I	.	.	I
		K	K	.	.	.	.	K
C	B	D	E	I	I	.	.	I
		K	K	.	.	.	.	K
D	E	I	I	.	.	.	.	I
E	E	I	I	.	.	.	.	I
F	D	G	G	.	.	.	.	G
		H	H	.	.	.	.	H

Such complex use of the aliases facility is not encouraged and is presented for illustrative purposes only.

Note: In particular, for most normal goals any particular entry should appear in either the `alias` file or the `alias` database, *not in both*!

3.1.2.2 Format of the Alias Database

The `alias` database has the same format as the optional domain database file described in Section 2.2.9. This means that aliases in the database are limited to 32 characters in length and can translate to a string containing at most 80 characters unless a "long" database is used. See Section 2.2.9 for information on long databases.

Length restrictions aside, `alias` database entries are handled in the same way as `alias` file entries and can be used in exactly the same way. Both multiple addresses and mailing list references are allowed. Both the `alias` file and `alias` database must be world readable.

The PMDF `alias` database is created from an input text file (*not* from the `alias` file—from a *different* input text file) using the `PMDF CRDB` (OpenVMS) or `pmdf crdb` (UNIX or NT) utility, described in Chapter 29 and Chapter 30. The format of entries in the input file for CRDB or `crdb` should be:

```
alias1   alias-value1
alias2   alias-value2
   .           .
   .           .
   .           .
```

Note that unlike the aliases file, the entries in the alias database source text file normally do not use a colon to separate the alias from its value.

On OpenVMS systems, the alias database should be generated with the commands:

```
$ PMDF CRDB input-file-spec PMDF_TABLE:aliases.tmp
$ RENAME PMDF_TABLE:aliases.tmp PMDF_ALIAS_DATABASE
```

An intermediate, temporary database is used so as to minimize any window of time during which the database file is in an undefined state as it is being generated or regenerated.

On UNIX systems, use the commands

```
# pmdf crdb input-file-spec PMDF_ALIAS_DATABASE
```

On NT systems, use the commands

```
C:\ pmdf crdb input-file-spec PMDF_ALIAS_DATABASE
```

Alternatively, a source file using colons, (that is, of the same format as the alias file), *e.g.*,

```
alias1:   alias-value1
alias2:   alias-value2
   .           .
   .           .
   .           .
```

may be used providing that the `/STRIP_COLONS` (OpenVMS) or `-strip_colons` (UNIX or NT) qualifier is used when building the database; *e.g.*, on OpenVMS:

```
$ PMDF CRDB/STRIP_COLONS input-file-spec PMDF_TABLE:aliases.tmp
$ RENAME PMDF_TABLE:aliases.tmp PMDF_ALIAS_DATABASE
```

or on UNIX:

```
# pmdf crdb -strip_colons input-file-spec PMDF_ALIAS_DATABASE
```

or on NT:

```
C:\ pmdf crdb -strip_colons input-file-spec PMDF_ALIAS_DATABASE
```

3.1.3 Personal Alias Databases (OpenVMS and UNIX)

Both the `alias` file and `alias` database are system-wide entities. They do not provide a mechanism that lets individual users set up personal aliases for single addresses or distribution lists.

On OpenVMS and UNIX platforms, PMDF provides an additional database facility which is user-accessible; each user can create and use his or her own database of addresses and lists. On OpenVMS, personal alias databases are consulted during both initial message submission by the owner of the personal alias database and delivery of messages to that user by the `L` channel; on UNIX, personal alias databases are consulted only during initial message submission by the owner of the personal alias database.

On OpenVMS, this database is located via the PMDF_PERSONAL_ALIAS_DATABASE logical name; this logical usually translates to SYS$LOGIN:aliases.dat. The use of SYS$LOGIN makes this database a per-user entity. Note that redefining the PMDF_PERSONAL_ALIAS_DATABASE logical on a per-user or per-group basis is not supported. In particular, PMDF can and does make use of personal aliases during local delivery. In doing this PMDF cannot have knowledge of any user-level changes to this logical name. As such PMDF simply expects to find personal alias databases in user login directories.

On UNIX, this database is located via the PMDF_PERSONAL_ALIAS_DATABASE tailor file option; this option is usually set to ~/aliasesdb. The use of the ~ initial path makes this database a per-user entity.

The format of the personal alias database is upwards compatible with the format of the system alias database. Some additional flag bits are defined which have specific meaning for user aliases. These flags are described fully in the documentation for the PMDF DB (OpenVMS) or pmdf db (UNIX) utility; see the appropriate edition of the *PMDF User's Guide*.

Use of personal alias databases can be disabled with the USE_PERSONAL_ ALIASES PMDF option.

Personal alias databases are consulted before the system alias database is consulted.

Personal alias databases are created and managed on OpenVMS using the ALIAS commands in PMDF MAIL or the PMDF DB utility, or on UNIX using the pmdf db utility. For details on creating and using personal alias databases, see the appropriate edition of the *PMDF User's Guide*.

3.1.4 Logical Name Table Aliases (OpenVMS)

The OpenVMS version of PMDF also has the ability to use aliases stored in logical name tables. This facility differs from similar facilities in VMS MAIL in that a separate set of logical name tables can be used and multi-valued logical names can be used to make an alias translate to multiple addresses. The name tables must be protected such that they are world readable.

This facility is enabled by setting the NAME_TABLE_NAME PMDF option to the name of the logical name table which contains the aliases. This name can be a logical name in the process or system directory that in turn specifies the table; this can be a search list if multiple tables are to be searched. If the NAME_TABLE_NAME option is not explicitly set, logical names are not used as a source of alias information.

Logical name table aliases are consulted after the personal alias database but before the system alias file or database.

As an example, suppose that a table named ALIAS_TABLE is to be used. The following DCL commands create the table and store two aliases in it. The aliases are gripes and help which translate, respectively, to system@example.com and consultants@example.com.

```
$ CREATE/NAME_TABLE/PROTECTION=(S:RWED,O:RWED,G:RE,W:RE) ALIAS_TABLE
$ DEFINE/TABLE=ALIAS_TABLE GRIPES "system@example.com"
$ DEFINE/TABLE=ALIAS_TABLE HELP   "consultants@example.com"
```

The NAME_TABLE_NAME option in the PMDF option file, PMDF_OPTION_FILE, should then be set to

```
NAME_TABLE_NAME=ALIAS_TABLE
```

Mail then sent to in%gripes (in%"gripes@example.com") or in%help (in%"help@example.com") from within VMS MAIL will then be sent, respectively, to system@example.com or consultants@example.com. Similarly, mail received from the network which is addressed to gripes@example.com or help@example.com will be properly handled.

3.1.5 Restrictions on Aliases

There are some important restrictions that should be observed when using aliases:

1. The addresses in the alias file or database should be formatted as pure RFC 822 addresses, *e.g., user@host*. Do not try to use DECnet or other routing conventions that you can get away with in the rewrite rules table. Not only may such things fail, they may not produce a visible error (see the next item). Source routes are the only exotica that are permitted.

2. Certain types of bogus addresses in a list alias will *not* generate a "bad address" return message. Specifically, if, for a given address in the list, the system name is illegal or there is a syntax error in the address specification, then the copy of the message to that address may be silently dropped and no one will be the wiser. If the mailing list file associated with an alias does not exist, then mail to the list itself may be dropped. However, errors in the mailbox part of the address (*e.g.,* "no such user") will be handled correctly.

   System managers should take care to test each list they set up to insure that all the recipient addresses are correct. The PMDF TEST/REWRITE/CHECK_EXPANSIONS (OpenVMS) or pmdf test -rewrite -check_expansions (UNIX or NT) utility provide a way to do this. Lists should be checked periodically and also whenever extensive changes are made.

3. PMDF reads the alias file only as each program using PMDF initializes itself. This means that if you are using a permanently resident server (such as the multi-threaded SMTP server or PMDF-LAN Lotus Notes channels) you should be sure to stop and restart the server each time the alias file or any of the files it includes is changed. (The PMDF RESTART (OpenVMS) or pmdf restart (UNIX and NT) utility provide a simple way to restart any such PMDF detached processes.) On the other hand, mailing list files referenced by the alias file are read and reread as needed, so servers need not be restarted when one of these files is changed.

4. The alias file is always read into memory in its entirety each time PMDF is used. All files included by the primary alias file are also loaded into memory. (Mailing list files are not loaded into memory.) The use of a huge alias file can eat up lots of memory. Liberal use of the mailing list reference operator, <, to reference long lists is recommended. Long lists of addresses coded directly into the alias file or any files

it includes should be avoided. Use of an `alias` database for large numbers of aliases is also recommended.

5. Be sure to observe the length restrictions associated with aliases. Aliases in the alias file can contain up to 60 characters. Aliases in the database can contain up to 32 characters in a short database, up to 80 characters in a long database, and up to 252 charactes in a huge database. In the alias file, the addresses to which aliases translate can contain up to 252 characters. In the case of a short database, the translation value can contain up to 80 characters; in the case of a long database the translation value can contain up to 256 characters; in the case of a huge database the translation value can contain up to 1024 characters. In some cases failing to observe length restrictions may lead to addresses being silently dropped from lists.

3.2 Directory Channels

The `directory` channel is similar in function to the alias file. The alias file is only used when the addressee is on the local system, or matches a channel marked with the `aliaslocal` channel keyword, while the `directory` channel provides aliasing for other systems or `pseudo` domains which your system manages. The `directory` channel also provides facilities for looking up aliases using mechanisms other than a PMDF database.

The `directory` channel is used to set up pseudo domains — systems which exist only in a logical sense. A `directory` channel is used to transform the mailbox names associated with such a pseudo domain into mailboxes on real systems. Such a scheme can be used to standardize naming conventions for groups of disparate systems.

The directory channel includes special handling for subaddresses, akin to that for the local channel; see Section 2.3.4.71 and Section 3.1.1.6.

The transformations applied by the directory channel can be derived from a number of different information sources:

- Databases can be used, with a separate one for each pseudo domain. These databases are in the same format as PMDF's domain and alias databases and are created with the `PMDF CRDB` (OpenVMS) or `pmdf crdb` (UNIX or NT) utility.
- `LDAP` and `X.500` look ups can be used, with a separate set of defaults for each pseudo domain.
- A database providing the names of ALL-IN-1 distribution list files can be used. Addresses are then tranformed into the contents of the corresponding ALL-IN-1 distribution list (OpenVMS only).
- `CCSO/qi/ph` look ups can be used, with a separate set of defaults for each pseudo domain.

Note that with appropriate use of the `aliaslocal` channel keyword, the alias database can be used to implement functionality similar to the directory channel's `crdb` database type of lookup on arbitrary pseudo domains. Similarly $(*text*) rewrite

rule substitutions and the PMDF general database[6] can be used to implement similar functionality. Such alias database or general database use avoids the overhead of additional channel processing incurred by the directory channel. But although such schemes may be more efficient than using a `directory` channel, the `directory` channel's `crdb` lookups do have some additional features such as support of duplicate usernames, support of alternate postmaster return addresses, and better diagnostic messages when illegal addresses are used.

3.2.1 Directory Channel Definition and Rewrite Rules

The first step in installing the `directory` channel is to add the channel entry towards the bottom of your PMDF configuration file. The entry should be of the form:

```
directory
DIRECTORY-DAEMON
```

Rewrite rules of the form

domain $U%*domain*@DIRECTORY-DAEMON

should also be added towards the top of your PMDF configuration file to map each pseudo domain *domain* supported onto the directory channel. For example, if `example.com` and `a1.example.com` are pseudo domains to be handled by the directory channel, rewrite rules of the form:

```
example.com             $U%example.com@DIRECTORY-DAEMON
a1.example.com          $U%a1.example.com@DIRECTORY-DAEMON
```

should be used.

More than one pseudo domain can be accommodated by the same directory channel; in general, there is no need to have separate channels for each domain. Simply add additional rewrite rules for any new pseudo domains and configure them as the following sections describe.

Alternatively, separate `directory_*` channels can be used for separate pseudo domains if desired, a typical reason being to allow specifying distinct postmaster return addresses for the different pseudo domains. In this case, define separate channels, each with their own rewrite rules, and use the `returnaddress` keyword on each channel to specity the desired postmaster address. For instance, channel definitions could be:

```
directory returnaddress postmaster@example.com
DIRECTORY-DAEMON

directory_a1 returnaddress postmaster@a1.example.com
DIRECTORY-A1
```

with rewrite rules

[6] On OpenVMS, the logical `PMDF_GENERAL_DATABASE` points to the general database, which is generally named `PMDF_TABLE:general.dat`; on UNIX, the option `PMDF_GENERAL_DATABASE` in the PMDF tailor file points to the general database, usually `/pmdf/table/generaldb.*`; on NT, the `PMDF_GENERAL_DATABASE` Registry entry points to the general database, usually `C:\pmdf\table\generaldb.*`.

```
example.com             $U%example.com@DIRECTORY-DAEMON
a1.example.com          $U%a1.example.com@DIRECTORY-A1
```

3.2.2 Directory Channel Inline Mode

Normally, look ups using the `directory` channel are queued to the directory channel itself and processed by the directory channel master program. When the channel keyword `inline` is specified on the `directory` channel, then look ups are done immediately, similar to the way look ups are done to the alias database. For example:

```
directory inline
DIRECTORY-DAEMON
```

Using inline directory lookups can help performance. Only local databases and LDAP look ups are supported in inline mode. Other directory sources are always queued to the `directory` channel.

A related option in the directory channel option file (see below) is `INLINE_AMBIGUOUS`, as described below:

INLINE_AMBIGUOUS (0 or 1)
If `INLINE_AMBIGUOUS` is set to 0, then the request is requeued to the `directory` channel so that the list of possible valid usernames can be sent back in a non-delivery notification. By default, ambiguous usernames are rejected immediately (INLINE_AMBIGUOUS=1).

The setting of `INLINE_AMBIGUOUS` does not affect the processing of invalid usernames or single valid usernames while in inline mode. Invalid names are rejected immediately, and valid single names are accepted immediately regardless.

3.2.3 Directory Channel Option File

The next step is to create a directory channel option file. The name of the option file is *x*`_option` where *x* is the name of the channel, hence usually `directory_option`, and the file should be placed in the PMDF table directory.

At a minimum, this file is used to tell the `directory` channel how to handle each pseudo domain it services. There should be at least one entry for each pseudo domain. These entries have the form:

domain=service-type

Here *domain* is the name of the pseudo domain in question and *service-type* is an integer indicating what sort of database is used to translate addresses in the pseudo domain:

Service type	Description
0	Use a PMDF CRDB or pmdf crdb database
2	Use an LDAP or X.500 directory database
3	Perform ALL-IN-1 list expansion operations
4	Use a CCSO/qi/ph directory database

The default for a pseudo domain if no option is specified is normally 0, a look up in a database created with PMDF CRDB (OpenVMS) or pmdf crdb (UNIX or NT). However, the option DEFAULT_METHOD may be used to change this default, as described below:

DEFAULT_METHOD (integer)
DEFAULT_METHOD may be set to any of the supported service types to select that service type as the default for pseudo domains which do not have an explicit setting. The default is DEFAULT_METHOD=0, meaning to use PMDF CRDB or pmdf crdb databases.

Continuing the previous example from Section 3.2.1, a sample directory channel option file might read:

```
example.com=0
a1.example.com=3
```

Note that additional transformation-specific options may be required in the directory channel option file. These options are described in the sections below.

3.2.4 Handling Multiple Pseudo Domains

A directory channel can service multiple pseudo domains. To apply a channel option to only a specific pseudo domain, prefix the option name with the name of the pseudo domain name followed by an underscore, _. See, for instance, the second example option file presented in Section 3.2.7.4. In that example, the LDAP_SERVERS option applies globally while the options prefixed with example.com_ apply only to the example.com pseudo domain. Likewise, the options prefixed with sales.example.com_ apply only to the sales.example.com pseudo domain.

An option setting prefixed with the pseudo domain name always takes precedence over a global option setting for the associated pseudo domain. For instance, in the option file

```
SIZELIMIT=20
a1.example.com_SIZELIMIT=10
```

the pseudo domain a1.example.com will use the value 10 for the SIZELIMIT option. All other domains will use the value 20 for that same option.

3.2.5 CRDB or crdb Database Operations

When using a PMDF database, (*i.e.*, a database created with PMDF's `CRDB` or `crdb` utility), on OpenVMS and UNIX you must create a directory to hold the database files. (On NT, the appropriate directory is created during the PMDF installation.) On OpenVMS systems, use the command:

```
$ CREATE/DIR pmdf_root:[directories]/OWNER=[SYSTEM]
```

On UNIX systems, use the commands

```
# mkdir -mu=rwx,go= /pmdf/directories
# chown pmdf /pmdf/directories
```

A separate database is needed for each pseudo domain. On OpenVMS, the database consists of a single database file whose name is derived from the pseudo domain name by replacing every period in the domain name with a dollar sign and appending "`.dat`"; on UNIX and NT, the database consists of several files whose name is the actual pseudo domain name with the appropriate file type appended. For example, if the pseudo domain name is `x.y`, the corresponding database file would be `pmdf_root:[directories]x$y.dat` on an OpenVMS system. On a UNIX system, the database would be represented by the files `/pmdf/directories/x.y.idx`, `/pmdf/directories/x.y.lck`, and `/pmdf/directories/x.y.pbl`. On an NT system, the database would be represented by the files `C:\pmdf\directories\x.y.idx`, `C:\pmdf\directories\x.y.lck`, and `C:\pmdf\directories\x.y.pbl`.

3.2.5.1 Database Entries

When using a PMDF database for a pseudo domain, each entry in the database consists of a mailbox name in the pseudo domain and the corresponding "real" address.

For example, suppose the pseudo domain is `example.com`, the one mailbox within this pseudo domain is `john.doe`, and the real address corresponding to this mailbox is `ariel@example.com`. To set up this domain, start with a text file containing the line:

```
john.doe    ariel@example.com
```

Then, on OpenVMS systems, assuming that the input text file is named `example$com.txt`, process this file with the `PMDF CRDB` utility as follows:

```
$ PMDF CRDB/DUPLICATES example$com.txt -
$_           pmdf_root:[directories]example$com.dat_tmp
$ RENAME pmdf_root:[directories]example$com.dat_tmp -
$_           pmdf_root:[directories]example$com.dat
```

An intermediate, temporary database is used so as to minimize any window of time during which the database file is in an undefined state as it is being generated or regenerated.

On UNIX systems, assuming that the input text file is named `example.com.txt`, use the commands

```
# pmdf crdb -duplicates example.com.txt /pmdf/directories/example.com
```

On NT systems, assuming that the input text file is named `example.com.txt`, use the commands

```
C:\> pmdf crdb -duplicates example.com.text \pmdf\directories\example.com
```

3.2.5.2 Default Entries

Special entries can be used to implement default redirections. Such entries are only used when no other entry matches the mailbox. The primary default rule has a single asterisk, `*`, as the mailbox:

```
*               *@host.domain
```

In this case the message is redirected to *host.domain* using the original mailbox specification. An entry of the form

```
*               newmailbox@host.domain
```

does the same thing except that *newmailbox* is used as the mailbox.

Two other special entries are available. The first is the special mailbox `*%*`, which matches any mailbox specification containing a percent sign. This is useful for matching and handling percent-routed addresses. The second special entry is `*!*`, which matches any mailbox containing an exclamation point. Both of these rules will be tried before the `*` rule is attempted.

3.2.5.3 Wildcard Entries

Other than the default entries listed above, the only other type of wildcard entries supported are ones where the subaddress is wildcarded. For example:

```
mailbox+*     newmailbox@host.domain
```

No other wildcarding of entries is supported.

3.2.5.4 Subaddresses

The way that directory channel lookups handles subaddresses is as follows. First, the entire address including the subaddress is looked up. Second, the subaddress is replaced with the wildcard (asterisk) and that is looked up. And finally the subaddress is stripped altogether and just the mailbox is looked up.

For example, if you have an address such as "jones+spam", the complete list of variants looked up in a directory channel database is as follows:

jones+spam
jones+*
jones

*

3.2.5.5 Duplicate Entries

Databases created with CRDB or `crdb` can contain duplicate mailboxes (if the `/DUPLICATES` or `-duplicates` qualifier is used). The `directory` channel uses this capability to provide an informative way of handling ambiguous addresses. An error message is returned when the mailbox extracted from an address matches a set of duplicate entries. The addresses associated with the duplicates should be unambiguous entries associated with the pseudo domain. This list of possible addresses is returned as part of the error message so the recipient of the error can select an appropriate address to use in future messages.

For example, suppose that the `example.com` pseudo domain is set up to contain entries for first names, last names, and dotted combinations of first names and last names. There are bound to be ambiguities in such a scheme for some common names. Specifically, suppose that entries for John Smith, Jane Smith, and John Jones are implemented. The entries for the names John and Smith would then be ambiguous. Therefore, instead of listing an actual address for these entries it would be more appropriate to list the unambiguous equivalents in the directory. This leads to a set of entries that might look like this:

```
john.smith          smithjo@vaxa.example.com
john                john.smith@example.com
smith               john.smith@example.com
jane.smith          smithja@vaxa.example.com
jane                smithja@vaxa.example.com
smith               jane.smith@example.com
john.jones          jj0u887@vaxb.example.com
john                john.jones@example.com
jones               jj0u887@vaxb.example.com
```

A message sent to `smith@example.com` would then produce an error message, but the message would recommend that either `john.smith@example.com` or `jane.smith@example.com` be used to disambiguate the address.

Note that PMDF contains no automatic facility to produce such databases; detection and resolution of ambiguities must be done by user-supplied programs.

3.2.6 ALL-IN-1 List Expansion Operations (OpenVMS)

When performing ALL-IN-1 list expansions, you must create a directory to hold the database files. Use the command:

```
$ CREATE/DIR pmdf_root:[directories]/OWNER=[SYSTEM]
```

A separate database is needed for each pseudo domain. The database names are derived by replacing every period in the domain name with a dollar sign and appending ".`dat`". For example, if the pseudo domain name is x.y, the corresponding database file would be `pmdf_root:[directories]x$y.dat` on an OpenVMS system.

The entries in the directory databases consist of a mailbox name in the pseudo domain and the name of the file containing the ALL-IN-1 list to be equated with that mailbox name. The directory channel looks up the mailbox name in the database, opens the associated list file, reads it and sends copies of the message to all of the addresses on the list. The database file itself should be created with the PMDF CRDB utility.

An example list file is shown in Example 3–4. Assume that the file specification for that file is d1:[lists]cats.dis and that this list is to be associated with the address cats-list@a1.example.com. Then the input test file to PMDF CRDB would appear as shown in Example 3–5. To process that input, use the commands:

```
$ PMDF CRDB a1$example$com.txt -
$_          pmdf_root:[directories]a1$example$com.dat_tmp
$ RENAME pmdf_root:[directories]a1$example$com.dat_tmp -
$_          pmdf_root:[directories]a1$example$com.dat
```

An intermediate, temporary database is used so as to minimize any window of time during which the database file is in an undefined state as it is being generated or regenerated.

Example 3–4 ALL-IN-1 Distribution List File d1:[lists]cats.dis

```
Carl Donner        ( DONNER-CD-O7814 )
Bob Smith          ( SU=Smith@GI=Bob@C=US@A=ATTMAIL@PRI=USDOE@O=HQ@X400@VENUS )
JUDY PUBLIC        ( SU=PUBLIC@GI=JUDY@C=US@A=ATTMAIL@PRI=USDOE@O=HQA@X400@VRY )
Remote Address     ( smj%sips.state.nc.us@PMDF@VENUS )
Remote Addressee   ( Joe%EXAMPLE.COM@PMDF@VENUS )
```

Example 3–5 Text File a1$example$com.txt **Used to Create ALL-IN-1 List Expansion Database**

```
cats-list    d1:[lists]cats.dis
```

3.2.7 LDAP or X.500 Directory Operations

When a service type of 2 is specified, the directory channel performs LDAP queries, querying an LDAP directory or X.500 DSA via an LDAP server, to look up mailbox names. The types of queries can be controlled with an LDAP filter file (*e.g.,* exact matches, fuzzy matches, searches down the entire directory, *etc.*). In the event of an ambiguous match, the possible choices can be returned along with the original message to the message originator.

The directory channel queries an LDAP directory or X.500 DSA via either a local or remote LDAP server. TCP/IP is used to communicate with the LDAP server; on OpenVMS systems UCX emulation is required of your TCP/IP package.

3.2.7.1 Required Options

When performing LDAP or X.500 directory look ups, the directory channel needs to know the LDAP server to which to connect and the point in the LDAP/X.500 hierarchy to which to bind and from which to base searches. Additional options, described in Section 3.2.7.3, may be used to control other aspects of the LDAP querying process.

3.2.7.1.1 LDAP_SERVERS Option

The LDAP_SERVERS option must be used to specify the LDAP server and port to which to connect. The format of this option is

LDAP_SERVERS=*host1*+*port1*|*host2*+*port2*|*host3*+*port3*...

At least one host must be specified. Hosts may be specified either by name, or by IP address. The port number may optionally be specified. Additional hosts and ports may optionally be specified; when multiple hosts are specified, they will be tried in left to right order.

The default port, if none is specified, is port 389—the standard port for LDAP servers. For example,

```
LDAP_SERVERS=ldap.example.com
```

or

```
LDAP_SERVERS=192.135.12.1
```

or

```
LDAP_SERVERS=ldap.example.com+389
```

3.2.7.1.2 LDAP_BASE Option

The LDAP_BASE option must also be specified in the option file. This option specifies the distinguished name of the location in the LDAP or X.500 directory information tree from which to base searches. The LDAP_BASE is specified in LDAP DN syntax according to RFC 1485; *e.g.,*

```
LDAP_BASE=o=Example Computing,st=California,c=US
```

3.2.7.2 TLS Options

PMDF has the ability to access LDAP servers using TLS authentication. Note that in order to use this feature, your LDAP server must be set up to do TLS. To configure the directory channel to use TLS, you must specify the following options.

TLS_MODE (1 or 2)

The TLS_MODE option is used to specify whether to use TLS. A value of 1 tells the directory channel to try to use TLS, but continue without it if TLS is not available. A value of 2 tells the directory channel to require TLS. The default is to not use TLS.

CACERTFILE (file name)

You may need to have the Certificate Authority (CA) certificate to be used by LDAP on your PMDF system. If so, by default the CA certificate should be placed in the file pmdf_table:ldap-cacert.pem. Use the CACERTFILE option if you wish to specify a different file name, for example if you need to use different CA certificates for different domains.

An example directory channel option file which includes TLS options is as follows:

```
example.org=2
example.org_ldap_servers=ldap.example.org
example.org_ldap_base=dc=ldap.example,dc=com
example.org_tls_mode=2
example.org_cacertfile=/pmdf/table/example-cacert.pem
```

3.2.7.3 Additional Options

Additional channel options are shown below:

BIND (0 or 1)

The BIND option is used to specify whether an LDAP bind operation is sent to the LDAP directory before a search operation is attempted. Unlike LDAPv2, LDAPv3 does not require a bind operation to take place. The default value is 1, meaning that a bind operation will be performed. If authentication is not required on the LDAPv3 server, performance can be improved by disabling bind operations by setting BIND=0. The BIND option is ignored when using the connectionless protocol over a UDP transport.

DISPLAY_MAIL_TYPE (attribute type)

When multiple matches to a search are found, the directory channel returns choices to the sender. The directory channel returns the distinguished name and e-mail address. By default, the e-mail address which the channel would use to deliver the mail (as specified by the MAIL_TYPE option) is displayed in such returns. The DISPLAY_MAIL_TYPE option can be used to specify an alternate mail address attribute to be returned. In particular, when redefining MAIL_TYPE to something other than the mail attribute, you may still want the directory channel to display the mail attribute when returning address choices to users. For example:

```
DISPLAY_MAIL_TYPE=mail
```

DISPLAY_MAIL_TYPE defaults to the value of MAIL_TYPE if not specified.

DN

The DN option is used to specify the LDAP or X.500 Distinguished Name used to bind to the LDAP directory or X.500 DSA — the DN is essentially the username to use to login to the server — although when using the directory channel over UDP transport, the DN is used by the LDAP server only for logging purposes and it is not passed to an X.500 DSA (if an X.500 DSA is backing up the LDAP server) during the bind. The DN is specified in LDAP DN syntax according to RFC 1485; *e.g.,*

```
DN=cn=Directory Channel,o=Example Computing,st=California,c=US
```

FILTERFILE

The directory channel processes each name by performing a regular expression match on the pmdf_dirchan group of rules in the ldapfilter.conf file in the PMDF table directory, *i.e.*, the file PMDF_TABLE:ldapfilter.conf (OpenVMS) or the file /pmdf/table/ldapfilter.conf (UNIX) or normally (the exact drive may be different depending upon installation) the file C:\pmdf\table\ldapfilter.conf (NT). Do not modify the supplied ldapfilter.conf file, as your changes will be lost when you upgrade or reinstall PMDF. Instead, to use a different file, specify the FILTERFILE option with the filename of the desired file. For example, on OpenVMS

```
FILTERFILE=PMDF_TABLE:example_ldapfilter.txt
```

or on UNIX

```
FILTERFILE=/pmdf/table/example_ldapfilter.txt
```

or on NT

```
FILTERFILE=C:\pmdf\table\example_ldapfilter.txt
```

The default ldapfilter.conf file contains a rich set of default rules which provide for exact and approximate matching of names and initials. However, if you want to make changes, see the comments in the file and Section 3.2.7.6 for details. The filters specified in this file are as defined in RFC 2254 (which obsoletes RFCs 1960 and 1558, the earlier descriptions of such filters).

FILTERTAG

The directory channel processes each name by performing a regular expression match on a group of rules found in the file specified by the FILTERFILE option. The FILTERTAG option is used to specify the group of rules to use. For example:

```
FILTERTAG=example_dirchan
```

The default is pmdf_dirchan. Do not modify the supplied ldapfilter.conf file, as your changes will be lost when you upgrade or reinstall PMDF. Instead, to use a different file, see the FILTERFILE option.

HINT_TYPE

When multiple matches to a search are found, the directory channel returns choices to the sender. In addition to the distinguished name and e-mail address, the directory channel can optionally return one more attribute from the entries to help the sender choose between them. For example,

```
HINT_TYPE=title
```

While any attribute can be specified, some suggestions are title, uid, telephoneNumber, or description.

LDAP_BASE (distinguished name)

The LDAP_BASE specifies the distinguished name of the location in the LDAP or X.500 directory information tree from which to base searches. See Section 3.2.7.1.2 for details.

LDAP_SERVERS (domain name or IP address)

The LDAP_SERVERS option is used to specify the IP address or domain name of the LDAP server to use. See Section 3.2.7.1.1 for details.

MAIL_TYPE (attribute type)

When the directory channel searches the LDAP directory or X.500 directory for a name, it requests that an e-mail address be returned. The MAIL_TYPE option is used to specify the attribute type requested from the directory. MAIL_TYPE must match the attribute type returned by your LDAP server; (while servers may accept aliases, they return one specific attribute type with the value). The default is MAIL_TYPE=mail. You may need to specify this option if you are using a non-PMDF LDAP server or you are using an LDAP or X.500 schema other than COSINE/Internet schema (RFC 1274). You will want to specify this option if you use a different directory attribute, such as pMDFMailAddress to specify a local delivery address. For example:

```
MAIL_TYPE=pMDFMailAddress
```

PASSWORD (string)

The PASSWORD option is used to specify a simple authentication credential to be sent with the DN (that specified by the DN option) when binding to the LDAP directory or X.500 DSA. This can be used to allow the directory channel more access to the directory than is allowed for anonymous users. For example:

```
PASSWORD=secret
```

If a PASSWORD is specified, a DN must also be specified, although a DN may be specified without a PASSWORD. The PASSWORD value is ignored when using the connectionless protocol over a UDP transport.

SIZELIMIT (integer >= -1)

When the directory channel performs a search for an e-mail address, many entries may match the search criteria. If this is the case, the original mail message is returned to the sender along with a list of possible address choices. The SIZELIMIT option controls the maximum number of choices which are returned; *e.g.,*

```
SIZELIMIT=10
```

The default value for SIZELIMIT is 50. You may want to make this limit smaller to reduce “trawling” of your database. Note that this limit may be superseded by a smaller limit which has been imposed by the manager of the LDAP directory or X.500 DSA.

Specify a value of -1 to allow any number of matches to be returned; specify a value of 0 to suppress the return of possible matches. Note that this is a change of behavior from versions of PMDF prior to V5.1-9 when a value of 0 allowed any number of matches to be returned.

TRANSPORT (TCP or UDP)

The TRANSPORT option is used to specify whether to use connection oriented LDAP protocol over TCP or connectionless oriented protocol over UDP. For example:

```
TRANSPORT=UDP
```

The default is TRANSPORT=TCP. When running over UDP, the slightly different CLDAP protocol is actually used. CLDAP is more suited for lower overhead over reliable network connections. Use LDAP over TCP if you may have packet loss to your server.

When using UDP, all information must fit in a single UDP datagram. If you use UDP, it is suggested that you specify a small SIZELIMIT option, *e.g.*, 10 or less. If the response from the LDAP server exceeds the size of a UDP datagram, you will not get any choices returned for ambiguous names.

TRIM (integer)

When multiple matches to a search are found, the `directory` channel returns to the sender a list of the matches. `TRIM` affects the level of detail provided in the returned information. If `TRIM` is a positive integer, it specifies how many elements to trim off of each matching distinguished name starting with the most general element and working down to the most specific element. A `TRIM` value of zero specifies that no trimming is to be done. A negative value specifies the number of elements to leave. For example, if the returned match is `Joe User, Accounting, Example Computing, California, US` then the following table shows the results of various `TRIM` values.

TRIM	Result
4	Joe User
3	Joe User, Accounting
0	Joe User, Accounting, Example Computing, California, US
-1	Joe User
-2	Joe User, Accounting

The default value of `TRIM` is `-1` so that only the most specific element is returned. A common choice for `TRIM` is the number of elements in your `LDAP_BASE` distinguished name.

3.2.7.4 Example Option Files

An example option file is shown below.

```
example.com=2
LDAP_SERVERS=ldap.example.com
LDAP_BASE=o=Example Computing,st=California,c=US
```

The `example.com=2` option specifies that LDAP directory or X.500 directory operations are to be done for the example.com pseudo domain. The LDAP server `ldap.example.com` is used; queries will begin at the position o=Example Computing, st=California, c=US in the LDAP or X.500 directory hierarchy.

Shown below is an example of an option file for a `directory` channel which services two different pseudo domains.

```
example.com=2
sales.example.com=2
LDAP_SERVERS=ldap.example.com
example.com_LDAP_BASE=o=Example Computing, st=California, c=US
example.com_TRIM=3
example.com_HINT_TYPE=title
sales.example.com_LDAP_BASE=ou=Sales, o=Example Computing, st=California, c=US
sales.example.com_TRIM=4
sales.example.com_HINT_TYPE=telephoneNumber
```

3.2.7.5 Default Mailbox Syntax Supported

The `ldapfilter.conf` file provided with PMDF supports a number of syntaxes. For the exact syntaxes supported, see the file itself, located in the PMDF table directory, and Section 3.2.7.6. Here are a few examples of syntaxes that are likely to match "Joe Wilson":

```
"Joe Wilson"@example.com
Joe_Wilson@example.com
Joe.Wilson@example.com
J.Wilson@example.com
Wilson@example.com
Wilsen@example.com
title=President@example.com
```

The last example is valid if the value of "title" in Joe Wilson's entry is "President".

3.2.7.6 LDAP Filter Configuration File, ldapfilter.conf

The file `ldapfilter.conf` contains information used by LDAP clients, *e.g.*, the PMDF directory channel doing an LDAP or X.500 directory lookup. Blank lines and lines that start with the hash character, `#`, are treated as comments and ignored. The configuration information consists of lines that contain one to five tokens. Tokens are separated by white space. Double quotes can be used to include white space inside a token, *e.g.*, "*text moretext*".

3.2.7.6.1 Filter Sets

The file consists of a sequence of one or more filter sets. A filter set begins with a line containing a single token called a tag. The tag is used by the client to select the filter set.

3.2.7.6.2 Filter Lists

A filter set consists of a sequence of one or more filter lists. The first line in a filter list must contain four or five tokens: The value pattern, the delimiter list, a filter template, a match description, and an optional search scope.

1. The value pattern is a regular expression that is matched against the search value passed by the client to select the filter list to be used.
2. The delimiter list is a list of characters (in the form of a single string) that are used to break the search value into distinct words.
3. The filter template is used to construct an LDAP filter. Standard LDAP filter format as specified in RFC 1960 (which obsoletes RFC 1558) is used.

   The filter template is similar in concept to a C language `printf` style format string. Everything is taken literally except for the character sequences:

Sequence	Description
%v	Substitute with entire search string value
%v$	Substitute with last word of search string value
%v*n*	Substitute word *n*; *n* is a single digit 1-9
%v*m*-*n*	Substitute words *m* through *n*
%v*m*-	Substitute word *m* through the last word

Words are numbered from 1 to 9, left to right.

4. The match description is used as information to describe the sort of LDAP search that took place. It should correctly complete both of the following phrases:

```
One "match description" match was found for...
```

and

```
Three "match description" matches were found for....
```

5. The search scope is optional, and should be one of base onelevel, or subtree. If search scope is not provided, the default is subtree.

The remaining lines of the filter list should each contain two or three tokens: A filter template, a match description and an optional search scope. The value pattern and delimiter list tokens are the same as previously specified.

3.2.7.6.3 Example LDAP Filter Configuration File

Example 3–6 shows a sample LDAP filter configuration file containing one filter set, pmdf_lookup, which contains three filter lists.

Example 3–6 Sample LDAP Filter File

```
# ldap filter file
#
pmdf_lookup
  "[0-9][0-9-]*"     " "    "(telephoneNumber=*%v)"    "phone number"

  "@"                " "    "(mail=%v)"                "email address"
                            "(mail=%v*)"               "start of email address"

  "="                " "    "%v"                       "arbitrary filter"
```

3.2.8 CCSO/ph/qi Directory Operations

When a service type of 4 is specified, the directory channel will query a CCSO directory to look up mailbox names.[7] The types of queries are controlled with the QUERY_METHOD_ channel options. In the event of an ambiguous match, the possible choices will be returned along with the original message to the message originator.

[7] Interactive OpenVMS users may use the pop-up CCSO addressing form to query CCSO directories from within VMS MAIL, PMDF MAIL, or DECwindows MAIL. See <REFERENCE>(HEAD1_CCSOFORMS) for details.

The directory channel queries a CCSO directory via a TCP/IP connection to either a local or remote qi server.

VMS

Alternatively, on OpenVMS platforms if you are using Bruce Tanner's qi implementation, then the channel can use a connectionless protocol to communicate directly with the CCSO database.[8] See <REFERENCE>(HEAD4_CCSOFORMCONNECTIONLESS) for details on configuring this. When using TCP/IP on OpenVMS, UCX emulation is required of your TCP/IP package.

When a single, unambiguous match is found for a pseudo domain mailbox name, the message is redirected to the address associated with the matching entry's e-mail field. The name of the e-mail field is given by the EMAIL_NAME_FIELD option. The email field should therefore be an internal address to which to forward the message. If the matching entry has no email field, then the message is returned to the originator with a note stating so.

If more than one match is found, then the message is returned to the originator along with a list of up to fifty possible matches. The SIZELIMIT option may be used to place a different limit on the number of matches returned. The list of matches will show, for each match, the values of the name and department fields. Different fields can be selected with the NAME_FIELD_NAME and DEPARTMENT_FIELD_NAME options. In addition, an email address to use for the match will be generated from the mail field and mail domain information generated by a qi siteinfo command. If the qi server does not return mailfield information then the address associated with the field named email will be returned. A different field can be selected with the PUBLIC_EMAIL_FIELD_NAME option.

As an example, suppose that a qi server's response to a siteinfo command is:

```
-200:0:version:VMS qi V2.12
-200:1:administrator:directory_manager@example.com
-200:2:mailfield:alias
-200:3:maildomain:example.com
200:Ok.
```

In this case the mail field name is "alias" and the mail domain name is example.com. For an entry with an alias field value of Bob.Smith, the generated email address would then be Bob.Smith@example.com.

If no matches are found then the message is returned with a note stating so.

[8] Copies of Bruce Tanner's implementation of qi and CCSO may be obtained from ftp.cerritos.edu; log in as anonymous and look in the vmsnet subdirectory for a qi*.zip file. Each qi*.zip file is a complete distribution distinguished by its version number.

3.2.8.1 Required Options

`Channel` options must be used to specify the qi server with which to communicate and to control aspects of the `CCSO` queries performed. To this end, the `QI_SERVERS` and one or more `QUERY_METHOD_` options must be specified in the `directory` channel's option file.

3.2.8.1.1 QI_SERVERS Option

The `QI_SERVERS` option specifies the TCP/IP host names of the qi servers to use. This is a required option; if it is not specified, the directory channel will not perform look ups.

The option's value takes the form, (note the use of the vertical bar character, `|`),

host1`+`*port1*`|`*host2*`+`*port2*`|`*host3*`+`*port3*`...`

where *host1*, *host2*, *host3*, ... and *port1*, *port2*, *port3*, ... are, respectively, the TCP/IP hosts and ports to which to connect. The hosts will be attempted in the order listed, from left to right, until a connection is successfully made to one of the hosts or the list is exhausted. IP addresses may be used in place of host names. If the port number is omitted then the standard qi port, port 105, will be used. When omitting the port number, also omit the +.

VMS

To bypass TCP/IP and communicate directly with an OpenVMS CCSO database, specify the hostname `<local-access>` (that exact string, including angle brackets). This is only supported on OpenVMS platforms and only in conjunction with Bruce Tanner's qi implementation for OpenVMS. See <REFERENCE>(HEAD4_CCSOFORMCONNECTIONL for further details.

For instance, to use the hosts `ph.example.com` and `admin.example.com` as qi servers, you can specify

```
QI_SERVERS=ph.example.com|admin.example.com+5000
```

Since the port number was omitted for `ph.example.com`, port 105 will be used. Port 5000 is used when connecting to `admin.example.com`.

3.2.8.1.2 QI_QUERY_METHOD_ Options

When attempting to locate an entry in a CCSO directory, the `directory` channel will generate up to ten qi `query` commands. The `query` commands will be tried one after the other until either a match (or matches) is found or the list of possible query methods is exhausted. The forms of the `query` commands are controlled by "query methods" specified in the option file. Since both the format of mailbox names and the behavior of qi `query` commands vary so widely, no default query methods are supplied by the `directory` channel. A set of one or more query methods must be specified.

Each query command will be of the form

```
query name-value return ...
```

where *name-value* is derived from the mailbox and pseudo domain name, and the ... portion of the command signifies the names of various CCSO directory fields which will be requested.

The method by which the *name-value* string is derived is controlled with the QUERY_METHOD_0, QUERY_METHOD_1, ..., QUERY_METHOD_9 options:

```
QUERY_METHOD_n=qi-command|name-format|translate-from-chars|translate-to-char
```

Here *qi-command* is an optional qi server command to issue prior to the query command (*e.g.,* set exact=on). *name-format* is a formatting string describing how to format the pseudo domain address for inclusion in the query command. *translate-from-chars* is an optional field specifying one or more characters which, when they appear in the mailbox name, will be translated to the optional character *translate-to-char*.

Table 3–2 describes control sequences which may be used in the *name-format* formatting string. Substitutions are done after any character translations have been performed.

Table 3–2 Query Method Formatting String Control Sequences

Sequence	Usage
%s	Substitute in the value of the mailbox name
%*s	Same as %s but with wild card, *, suffixes added to each blank delimited part of the field value; if LEADING_WILDCARDS=1 is specified in the option file then wild card prefixes will also be added
%h	Substitute in the pseudo domain name

To specify a literal %, |, or \, specify \%, \|, or \\.

As an example, let us assume that the address John.Doe@example.com is to be looked up in a CCSO database. In this case, the mailbox name is John.Doe and the pseudo domain name is example.com. Under this assumption, Table 3–3 shows sample query methods and the query commands that would result. In the table, the *qi-command* portion of the query method has been omitted.

Table 3–3 Example Formatting Strings

Formatting string	Resulting query command
|%s	query John.Doe return ...
|name=%s|_.|	query name=John Doe return ...
|%*s	query John.Doe* return ...
name=%*s|_.|	query name=John* Doe* return ...

In the second and fourth examples a space character follows the final |. Thus, the characters . and _ are changed to a space wherever they appear in a mailbox name.

To continue the above example, suppose that the three query methods shown below are specified:

```
QUERY_METHOD_0=set approx=off soundex=off|email=%s@%h
QUERY_METHOD_1=set approx=on|%s|_.|
QUERY_METHOD_2=|%*s|_.|
```

With these settings, a look up for `John.Doe@example.com` would result in the following sequence of `qi` commands.

```
set approx=off soundex=off
query email=John.Doe@example.com return ...
set approx=on
query John Doe return ...
query John* Doe* return ...
```

These commands will be tried one after the other until either a match is returned or the list of methods exhausted.

3.2.8.2 Additional Options

Additional options are listed below:

DEPARTMENT_FIELD_NAME (text string <= 251 characters long)
Specify the name of the `CCSO` directory's department field. This should be the name of the field giving the department name or other useful identifying information associated with a directory entry. This field is only used in the event of multiple matches. If not specified, the field name "department" is assumed. See Section 3.2.8 for futher details.

EMAIL_FIELD_NAME (text string <= 251 characters long)
Specify the name of the `CCSO` directory's email field. This should be the field with the "internal" email address to use for forwarding mail to an individual in the database. If not specified, the field name "email" is assumed. See Section 3.2.8 for further details.

LEADING_WILDCARDS (0 or 1)
Specifies whether or not leading wild cards, `*`, are put into strings formatted with the `%*s` control sequence. By default, `LEADING_WILDCARDS=0`, a leading wild card is not put into such strings since that reduces the efficiency of the look up process with some qi servers. Specify a value of 1 to have leading wild cards added.

NAME_FIELD_NAME (text string <= 251 characters long)
Specify the name of the `CCSO` directory's name field. This should be the name of the field giving the name associated with a directory entry. This name is only used when rejecting a message either because the matching database entry lacks an email field or in the event of multiple matches. If not specified, the field name "name" is assumed. See Section 3.2.8 for further details.

NO_MATCH_HOST (text string <= 252 characters long)
Specify a host to redirect messages to when no matches can be found in the `CCSO` directory. For instance, if

```
NO_MATCH_HOST=hub.example.com
```

and no match can be found for the user `Nerble`, then the message will be forwarded on to `Nerble@hub.example.com` rather than return the message as undeliverable.

PUBLIC_EMAIL_FIELD_NAME (text string <= 251 characters long)
Specify the name of the CCSO directory's public e-mail field. This should be the field with the "public" e-mail address to use as a hint when the address to be looked up is ambiguous. This field will only be used when information from the siteinfo command cannot be used. See Section 3.2.8 for further details. If not specified, the field name "email" is used.

QI_SERVERS (text string <= 252 characters long)
Specify the qi servers to contact. See Section 3.2.8.1.1 for details.

QUERY_METHOD_0, ..., QUERY_METHOD_9 (text string <= 252 characters long)
Specify the query methods to use. See Section 3.2.8.1.2 for details.

RECV_TIMEOUT (integer >= 0 seconds)
This option controls how long the directory channel will wait for a response from the qi server before giving up (timing out). By default, the channel will wait 120 seconds (RECV_TIMEOUT=120). To disable the timeout mechanism, specify RECV_TIMEOUT=0. This will cause the channel to wait indefinitely.

When a timeout occurs, the channel closes its connection to the qi server. When necessary, the channel will attempt to reestablish a connection with a qi server.

SITEINFO (0 or 1)
In the case of an ambiguous address, the message is bounced back to the sender with hints as to how to resolve the ambiguity. These hints will include the e-mail addresses of the possible matches. When SITEINFO=1, the default, the e-mail addresses will, if possible, be constructed from information gathered with the siteinfo command. When SITEINFO=0, the addresses will be the value, if any, of the email address field for each possible match. The PUBLIC_EMAIL_FIELD_NAME option specifies the name of the field to use for this purpose.

SIZELIMIT (integer >= -1)
When the directory channel performs a search for an e-mail address, many entries may match the search criteria. If this is the case, the mail is returned to the sender along with a list of possible choices. The SIZELIMIT option controls the maximum number of choices which are returned; *e.g.,*

```
SIZELIMIT=10
```

The default value for SIZELIMIT is 50. You may want to make this limit smaller to reduce "trawling" of your database. Note that this limit may be superseded by a smaller limit which has been imposed by the manager of the qi server or CCSO directory.

Specify a value of -1 to allow any number of matches to be returned; specify a value of 0 to suppress the return of possible matches.

STRIP_QUOTES (0 or 1)
Controls whether or not outer quotes are stripped from the local part of an address to be looked up. By default, STRIP_QUOTES=1, quotes are stripped. Thus, for the address

```
"Bob Smith"@example.com
```

query commands of the form

```
query Bob Smith return ...
```

would be generated. Were STRIP_QUOTES=0 specified, then the queries would be of the form

```
query "Bob Smith" return ...
```

which may not be appropriate.

3.2.8.3 Example Option Files

An example option file is shown below.

```
example.com=4
QI_SERVERS=qi.example.com|vaxc.example.com+5200
QUERY_METHOD_0=set exact=on|alias=%s
QUERY_METHOD_1=set approx=on|%*s
```

The example.com=4 option specifies that CCSO directory operations are to be done for the example.com pseudo domain. The QI server qi.example.com is used. If it is not reachable, then the qi server on port 5200 of vaxc.example.com will be used.

Shown below is an example of an option file for a directory channel which services two different pseudo domains.

```
stateu.edu=4
students.stateu.edu=4
QI_SERVERS=ph.athena.stateu.edu
stateu.edu_SIZELIMIT=10
students.stateu.edu_SIZELIMIT=15
students.stateu.edu_DEPARTMENT_FIELD_NAME=school
QUERY_METHOD_0=set exact=on|alias=%s
QUERY_METHOD_1=set approx=on|%*s
```

3.3 Address Reversal

Header From: addresses and other backwards-pointing addresses and forwards-pointing header addresses receive one additional processing step. This additional processing step is referred to as *address reversal*, and can be performed via LDAP lookups, and/or via use of a reverse database and/or REVERSE mapping. Section 3.3.1 discusses use of LDAP lookups for address reversal; Section 3.3.2 discusses use of the reverse database and REVERSE mapping. An LDAP lookup, if any, is performed prior to checking the reverse database and/or REVERSE mapping.

3.3.1 LDAP Lookups for Address Reversal

If the PMDF options LDAP_HOST, LDAP_PORT, and REVERSE_URL are specified in the PMDF option file (see Section 7.3.2), then each address passing through PMDF will be checked against the LDAP server specified by the LDAP_HOST and LDAP_PORT options, using the LDAP query specified by the REVERSE_URL option. If the LDAP query succeeds and returns a value, that value will be substituted in place of the original address.

For the REVERSE_URL option, standard LDAP URLs must be used, except with the host and port omitted, as the host and port are instead specified via the LDAP_HOST and LDAP_PORT PMDF options. That is, the LDAP URL should be specified as

ldap:///*dn*[?*attributes*[?*scope*?*filter*]]

where the square bracket characters *[* and *]* shown above indicate optional portions of the URL. The *dn* is required and is a distinguished name specifying the search base; it might correspond to the organization's top level in the Directory Information Tree, or it might correspond to a subset of the organization, based upon the domain name in the original address. The optional *attributes*, *scope*, and *filter* portions of the URL further refine what information to return. For address reversal, the desired *attributes* to specify returning would typically be the mail attribute (or some similar attribute). The *scope* may be any of base (the default), one, or sub. And the desired *filter* would typically be based upon the mailbox (local portion) of the incoming addresses. Certain substitution sequences may be used to construct the LDAP search URL; see Table 3–1 for details.

3.3.2 The Address Reversal Database and REVERSE Mapping

Header From: addresses and other backwards-pointing addresses and forwards-pointing header addresses receive one additional processing step.[9] PMDF uses each address specification with any routing address but less any personal name fields as an index key to a special database called the address reversal database.

The address reversal database must be world readable and is generally located in the PMDF table directory.[a] This database file is built with the PMDF CRDB (OpenVMS) or pmdf crdb (UNIX and NT) utility.

If the address is found in the database, the corresponding right hand side from the database is substituted for the address. If the address is not found an attempt is made to locate a mapping table named REVERSE in the mapping file. No substitution is made and rewriting terminates normally if the table does not exist or no entries from the table match.

[9] This processing can be restricted to only backwards pointing addresses if the third bit, bit 2, in the PMDF option USE_REVERSE_DATABASE is cleared; see Section 7.2.

[a] On OpenVMS systems the database is the file pointed at by the PMDF_REVERSE_DATABASE logical, which by default points to the file PMDF_TABLE:reverse.dat. On UNIX systems, the database is the file pointed to by the PMDF Tailor file option PMDF_REVERSE_DATABASE, by default /pmdf/table/reversedb.*. On NT systems, the database is the file pointed at by the PMDF_REVERSE_DATABASE Registry entry, generally C:\pmdf\table\reversedb.*.

Note 1: You do not need to have an address reversal database in order to use a REVERSE mapping. That is, you can use a REVERSE mapping without having an address reversal database. And, of course, the reverse is true: you do not need to have a REVERSE mapping to use an address reversal database.

Note 2: If you have a compiled configuration, then you must recompile and reinstall your configuration in order for changes to the REVERSE mapping table, (or indeed for any changes to the PMDF mappings file), to take effect.

If the address matches a mapping entry, the result of the mapping is tested. The resulting string will replace the address if the entry specifies a $Y; a $N will discard the result of the mapping. If the mapping entry specifies $D in addition to $Y, the resulting string will be run through the reversal database once more, and if a match occurs the template from the database will replace the mapping result (and hence the address). See Table 3–4 for a description of additional flags available for the REVERSE mapping, and Table 5–2 for a list of general mapping table substitution sequences and metacharacters.

Table 3–4 REVERSE Mapping Table Flags

Flags	Description
$Y	Use output as new address
$N	Address remains unchanged
$D	Run output through the reversal database
$A	Add pattern as reverse database entry
$F	Add pattern as forward database entry

Flag comparisons	Description
$:B	Match only header (body) addresses
$:E	Match only envelope addresses
$:F	Match only forward pointing addresses
$:R	Match only backwards pointing addresses
$:I	Match only message-ids

The reverse and noreverse channel keywords, and the PMDF options USE_REVERSE_DATABASE and REVERSE_ENVELOPE may be used to control the specifics of when and how address reversal is applied. In particular, address reversal will not be applied to addresses in messages when the destination channel is marked with the noreverse keyword. If USE_REVERSE_DATABASE is set to 0, address reversal will not be used with any channel. The REVERSE_ENVELOPE option controls whether or not address reversal is applied to envelope From: addresses as well as message header addresses. See the descriptions of these options and keywords for additional information on their effects.

The primary use of address reversal is to substitute a generic address (perhaps an address on a central machine) for addresses on remote, and possibly transitory, systems. Address reversal is a particularly powerful tool when used in conjunction with aliases or the directory channel.

Entries in the address reversal database consist of two e-mail addresses: the address to match against and the address with which to replace a match. The database is usually created by preparing a text file and processing it with the PMDF CRDB (OpenVMS) or pmdf crdb (UNIX or NT) utility.

For example, suppose a site wants to replace all reverse pointing addresses of the form *user*@example.com with an address of the form *first.last*@example.com where *first.last* is formed from the first (given) and last (family) names of the owner of the account USER. This will then cause the outside world to only see addresses of the form *first.last*@example.com and never see internal addresses. A text file reverse.txt containing lines of the form

Note: The exact format needed for reverse database entries is determined by the value of the option USE_REVERSE_DATABASE in the OPTION.DAT file.

```
user1@example.com  first1.last1@example.com
user2@example.com  first2.last2@example.com
     .                      .
     .                      .
     .                      .
```

should then be set up and converted to an address reversal database with the OpenVMS commands,

```
$ PMDF CRDB reverse.txt PMDF_TABLE:reverse.tmp
$ RENAME PMDF_TABLE:reverse.tmp PMDF_REVERSE_DATABASE
```

An intermediate, temporary database is used so as to minimize any window of time during which the database file is in an undefined state as it is being generated or regenerated.

or the UNIX commands,

```
# pmdf crdb reverse.txt /pmdf/table/reversedb
```

As another example, suppose that the internal addresses at example.com are actually of the form user@host.example.com, but, fortunately, the username space is such that user@hosta.example.com and user@hostb.example.com specify the same person for all hosts at example.com. Then, rather than have to enter all possible user and host combinations in the address reversal database, the following, very simple REVERSE mapping may be used in conjunction with the address reversal database:

```
REVERSE
  *@*.example.com           $0@example.com$Y$D
```

This mapping maps addresses of the form *user*@host.example.com to *user*@example.com. The $D flag causes the address reversal database to then be consulted. The address reversal database should contain entries of the form shown in the previous example.

Additional examples are given in Sections 3.4 and 3.6.

Normally a REVERSE mapping or reverse database must be constructed manually. The design and maintenance of such a database is likely to be a very site-specific task. The database, for the most part, may end up being the inverse of the translations imposed by aliases on the local system. This is not a requirement, however, and it may be

useful to have the database perform other, nonbijective (*i.e.,* non-invertible), address transformations.

Although there is no address reversal database or mapping table by default, address reversal is activated automatically once such an address reversal database or REVERSE mapping exists. (Note that if you have a compiled PMDF configuration, then you must recompile—and on OpenVMS reinstall—your PMDF configuration in order for changes to the REVERSE mapping to take effect.)

3.4 The Forward Database and FORWARD Address Mapping

Address reversals are not applied to envelope To: addresses. The reasons for this omission are fairly obvious — envelope To: addresses are continuously rewritten and modified as messages proceed through the mail system. The entire goal of routing is to convert envelope To: addresses to increasingly system and mailbox-specific formats. The canonicalization functions of address reversal are entirely inappropriate for envelope To: addresses.

In any case, plenty of machinery is available in PMDF to perform substitutions on envelope To: addresses. The alias file, alias database, and general database, as well as the directory channel and its associated databases, provide precisely this functionality.

PMDF also has available the forward database and FORWARD mapping, used for special sorts of forwarding purposes, such as pattern based forwarding, source-specific forwarding, or "autoregistration" of addresses. Note that the forward database and FORWARD mapping are intended for use primarily for certain special sorts of address forwarding; most sorts of address forwarding, however, are better performed using one of PMDF's other forwarding mechanisms. See Section 3.5 for an overview of various forwarding techniques.

The various substitution mechanisms for envelope To: addresses discussed previously provide functionality equivalent to the reversal database, but none yet discussed provide functionality equivalent to the reverse mapping. And circumstances do arise where mapping functionality for envelope To: addresses is useful and desirable.

The FORWARD mapping table provides this functionality of pattern based forwarding, and also provides a mechanism for source specific forwarding. If a FORWARD mapping table exists in the mapping file, it is applied to each envelope To: address. No changes are made if this mapping does not exist or no entries in the mapping match. See Chapter 5 for details on the mapping file.

If the address matches a mapping entry, the result of the mapping is tested. The resulting string will replace the envelope To: address if the entry specifies a $Y; a $N will discard the result of the mapping. See Table 3–5 for a list of additional flags.

The FORWARD mapping, if present, is consulted before any forward database lookup. If a FORWARD mapping matches and has the flag $G, then the result of the FORWARD mapping will be checked against the forward database, if forward database use has been enabled via the appropriate setting of USE_FORWARD_DATABASE. (Note that if channel specific forward database use has been specified, then the source address and source

channel will be prefixed to the result of the FORWARD mapping before looking up in the forward database.) If a matching FORWARD mapping entry specifies $D, then the result of the FORWARD mapping (and optional forward database lookup) will be run through the PMDF address rewriting process again. If a matching FORWARD mapping entry specifies $H, then no further FORWARD mapping or database lookups will be performed during that subsequent address rewriting (that resulting from the use of $D).

Table 3–5 FORWARD Mapping Table Flags

Flags	Description
$Y	Use output as new address
$N	Address remains unchanged
$D	Run output through the rewriting process again
$G	Run output through the forward database, if forward database use has been enabled
$H	Disable further forward database or FORWARD mapping lookups
$I	Hold the message as a .HELD file

Example 3–7 illustrates the use of a complex REVERSE and FORWARD mapping. Suppose that a system or pseudo domain named am.sample.example.com associated with the mr_local channel produces RFC 822 addresses of the general form:

"*lastname, firstname*"@am.sample.example.com

or

"*lastname,firstname*"@am.sample.example.com

Although these addresses are perfectly legal they often confuse other mailers which do not fully comply with RFC 822 syntax rules — mailers which do not handle quoted addresses properly, for instance. Consequently, an address format which does not require quoting tends to operate with more mailers. One such format is

firstname.lastname@am.sample.example.com

Example 3–7 A Complex FORWARD and REVERSE Mapping Example

```
REVERSE

  *|mr_local|"*,$ *"@am.sample.example.com $Y"$1,$ $2"@am.sample.example.com
  *|mr_local|"*,*"@am.sample.example.com   $Y"$1,$ $2"@am.sample.example.com
  *|*|"*,$ *"@am.sample.example.com        $Y$3.$2@am.sample.example.com
  *|*|"*,*"@am.sample.example.com          $Y$3.$2@am.sample.example.com
  *|mr_local|*.*@am.sample.example.com     $Y"$2,$ $1"@am.sample.example.com
  *|*|*.*@am.sample.example.com            $Y$2.$3@am.sample.example.com

FORWARD

  "*,$ *"@am.sample.example.com            $Y"$0,$ $1"@am.sample.example.com
  "*,*"@am.sample.example.com              $Y"$0,$ $1"@am.sample.example.com
  *.*@am.sample.example.com                $Y"$1,$ $0"@am.sample.example.com
```

So the goals of the sample mapping tables in Example 3–7 are threefold. (1) Allow any of these three address formats above to be used. (2) Present only addresses in the

original format to the mr_local channel, converting formats as necessary. (3) Present only addresses in the new unquoted format to all other channels, converting formats as necessary. (The REVERSE mapping shown assumes that bit 3 in the PMDF option USE_REVERSE_DATABASE is set; see Section 7.2.)

In cases where address forwardings need to be autoregistered, or source specific, the forward database is available. Note that use of the forward database for simple forwarding of messages is generally not appropriate; the alias database is a more efficient way to perform such forwarding. By default, the forward database is not used at all; its use must be explicitly enabled via the USE_FORWARD_DATABASE option, as described in Section 7.3.1. Forward database lookups are performed after address rewriting and after alias expansion is performed, and after any FORWARD mapping is checked. If a forward database lookup succeeds, the resulting substituted address is then run through the PMDF address rewriting process all over again.

The forward database is a PMDF crdb database, created using the PMDF CRDB (OpenVMS) or pmdf crdb (UNIX or NT) utility from a source text file. The format of the source text file by default is expected to be:

```
user1@domain1    changedmailbox1@changeddomain1
user2@domain2    changedmailbox@changeddomain2
```

But if source specific use of the forward database has been enabled by setting bit 3 of the USE_FORWARD_DATABASE option, then the source text file format expected is:

source-channel|*source-address*|*original-address* *changed-address*

For instance, an entry such as

```
snads_local|Bobby@BLUE|12345678@PMDF    user.with.a.long.name@else.where.com
```

would allow the Bobby@BLUE SNADS user to send to a SNADS address of 12345678@PMDF when they want to send to user.with.a.long.name@else.where.com.

3.5 Forwarding Mail

PMDF provides several mechanisms for forwarding mail. The method appropriate to a task at hand depends upon the scope of the forwarding:

Forwarding mail for selected users. To forward mail for selected users, it is best to use aliases. You may also use aliases to accept mail for a non-existent user and forward it on to one or more real users. See Section 3.5.1.

Forwarding mail to a list of users. Aliases are also used to create mailing lists. See Section 4.1; Section 4.1.3 contains a mailing list example.

Forwarding mail for selected users in other than the local domain. To forward mail for selected users in an arbitrary domain (a domain other than the local channel name), use of the directory channel is often appropriate. See Section 3.2.

Forwarding all mail for a given host to another host. In this case there are several approaches. The most efficient method requires that you be able to blindly change *user@old-host* into *user@new-host* without any conflict in user names; *i.e.,* not have to worry that the username "*user*" on old-host conflicts with a different person on *new-host* who has the same username. When this is the case, simple rewrite rules in the PMDF configuration file may be used. The less efficient, but just as effective, approaches involve using either a FORWARD mapping or directory channel. See Section 3.5.2.

3.5.1 Forwarding Mail for Selected Users

To forward mail for a few selected users, use the alias file. If, however, you will be doing this for several thousand users, then use the alias database. The difference between the two is that the alias file is loaded into memory while the alias database, intended for large numbers of aliases, is not. The alias database is a keyed, indexed file and has excellent performance even when it contains over a hundred thousand aliases.

VMS

On OpenVMS systems, the database is an RMS keyed, indexed file, and may be tuned with RMS tuning tools, if necessary.

Suppose you want to forward mail for the two local users Judy.Public and jdoe to, respectively, pjudy@vaxa.example.com and johndoe%a1@mr.xyzzy.org. To do this with the alias file you would simply add to the PMDF alias file, the two entries

```
Judy.Public:        pjudy@vaxa.example.com
jdoe:               johndoe%a1@mr.xyzzy.org
```

If you are using a compiled configuration, then be sure to recompile (and on OpenVMS reinstall) it after making this change.

To add these two entries to your alias database you should locate the source file used to generate that database, add these two entries to that file, and then regenerate the database with the PMDF CRDB (OpenVMS) or pmdf crdb (UNIX and NT) utility. There is no need to recompile your configuration after making changes to the alias database. However, resident servers (such as the PMDF SMTP server) may need to be restarted if you want them to see the change immediately.

On OpenVMS or UNIX systems, you may instead add these entries to the alias database using the PMDF DB (OpenVMS) or pmdf db (UNIX) utility. See the appropriate edition of the *PMDF User's Guide* for more details.

There are several points that you should note:

- PMDF's aliases only affect mail passing through PMDF. For instance, in the above example, the jdoe alias will have no effect for mail sent with VMS MAIL to JDOE unless you also forward JDOE's mail to IN%JDOE:

```
$ MAIL
MAIL> SET FORWARD/USER=JDOE IN%JDOE
MAIL> EXIT
$
```

This forwarding causes mail sent to JDOE to be handled by PMDF. This then allows PMDF's alias for jdoe to then take effect.

- Aliases are case insensitive. The alias for `Judy.Public` will match `judy.public`, `JUDY.PUBLIC`, `JuDy.PUBlic`, *etc.*
- Aliases only apply to local addresses, or to addresses matching a channel marked with the `aliaslocal` channel keyword. They do not apply to arbitrary addresses, such as in messages passing through your system for a different host (unless that host is handled by a PMDF channel marked `aliaslocal`).
- Aliases only affect the envelope `To:` address. The message header will not be rewritten; *i.e.,* the alias will not be expanded into the message header.
- In a similar vein to the previous item, messages passing through your system with an address such as `johndoe%a1@xyzzy.org` will not have that address magically replaced with the alias `jdoe@`*`local-host`*. The ability to do this is, however, very useful and may be accomplished with the address reversal database or `REVERSE` mapping. If you want to do this, then refer to Section 3.6.

3.5.2 Forwarding All Mail for a Host

When attempting to forward all mail for one or more hosts to another host, you must take into account mailbox name conflicts. Two sorts of conflicts may arise. First, when forwarding mail for the host *`old-host-1`* to the host *`new-host`*, you must ascertain whether or not there are mailbox names on *`old-host-1`* which also exist on *`new-host`* but correspond to different users. For example, suppose you forward mail for `bob@`*`old-host-1`* to `bob@`*`new-host`*. Have you just now sent Bob Smith's mail (`bob@`*`old-host-1`*) to Bilbo O. Baggin's mailbox (`bob@`*`new-host`*)? The second potential conflict arises when forwarding both mail for *`old-host-1`* and *`old-host-2`* to *`new-host`*: are there conflicts between mailbox names on *`old-host-1`* and *`old-host-2`*?

After you determine what sort of conflicts may arise, you can go ahead and set up the appropriate form of forwarding. If there are no mailbox name conflicts *and* the mailbox names remain unchanged, then you can use domain rewrite rules as described in Section 3.5.2.1. That is, if you will simply be forwarding all mail for *`user`*`@`*`old-host-1`*, *`user`*`@`*`old-host-2`*, ... *etc.*, to *`user`*`@`*`new-host`* with no change in the *`user`* part, then you can use rewrite rules. This is the most efficient and straightforward method. If there are conflicts or the mailbox name does not remain unchanged, then you will have to use either a `FORWARD` mapping, alias database, or directory channel. Use of the `FORWARD` mapping is preferable when you can algorithmically map the incoming address to its forwarding address, *e.g.*, if addresses of the form *`First.Last`*`@example.com` map to *`Last`*`@host.example.com`. If, however, you cannot specify a simple algorithm, then you should use either the alias database or a directory channel. Use the alias database when the incoming addresses are local addresses (*e.g.,* *`old-host-1`* is the local host). Otherwise, use the directory channel—or in some cases, use of a special channel marked with `aliaslocal` may be appropriate. The directory channel has the least optimal performance of all the options as it entails the use of an extra channel processing step.

See Section 3.5.2.2 for an example in which the FORWARD mapping is used; for information on the alias database or directory channel, see, respectively, Sections 3.1.2 or 3.2.

3.5.2.1 Using Rewrite Rules to Forward Mail

Suppose you want to forward all mail for *old-host-1* and *old-host-2* to *new-host* leaving the mailbox portion of the address unchanged. Then, to the upper portion of the PMDF.CNF file you would add the two rewrite rules

```
old-host-1          $U%new-host$E$F
old-host-2          $U%new-host$E$F
```

The EF causes these rewrite rules to only affect envelope To: addresses. It's that easy! Well, not quite. You should also consider whether or not you want to "transparently" forward these messages. The example shown above causes the forwarding to be transparent in the sense that no occurrence of *user@old-host-1* in the message header will be changed. If you want those occurrences to be changed also, then remove the $E from the rewrite rules. Moreover, if you want *user@old-host-1* to always be changed to *user@new-host* regardless of whether or not the address in question is forward (*e.g.,* To: or cc:) or backward pointing (*e.g.,* From:), then omit the $F.

One disadvantage to using rewrite rules to do this is that it often requires identifying and listing each old host name. If that is not feasible and you can identify the old host names via pattern matching, then use the FORWARD mapping as described in Section 3.5.2.2.

3.5.2.2 Using the FORWARD Mapping to Forward Mail

The FORWARD mapping is normally used to make pattern based, cosmetic changes to addresses after a message's envelope To: address has been rewritten and the destination of the message determined. However, with the FORWARD mapping's $D flag, it is possible to start the rewriting process anew using the output of the FORWARD mapping. That is, the FORWARD mapping may be used to alter an envelope To: address and then, using that altered address, redirect where the message should go. Again, note that the FORWARD mapping is used when the changes can be described in a pattern based, algorithmic fashion. If that is not possible then you will have to use either the alias file or database or a directory channel.

For instance, suppose that the following forwardings need to be effected:

```
First.Last@example.com      →
 LastF@example.com  (no change in destination host)
First_Last@example.com      →
 LastF@example.com  (no change in destination host)
"First Last"@example.com    →
 LastF@example.com  (no change in destination host)
First.Last@MR.example.com   →  "Last, First"%A1@MR.VAXA.example.com
First_Last@MR.example.com   →  "Last, First"%A1@MR.VAXA.example.com
"First Last"@Mr.example.com →  "Last, First"%A1@MR.VAXA.example.com
*-LIST@Obsolete.example.com →  *-L@Listserv.example.com
*%vax*@VAXA.example.com     →  *@VAX*.example.com
```

These forwardings may be realized with the FORWARD mapping table shown in Example 3–8. The following items of note are identified with callouts in that example.

❶ The first three entries do not require a $D flag as there is no change in the host name portion of the address.

❷ In the left-hand column a "$ " sequence is used to represent a literal space.

❸ This and subsequent entries do involve a change in host name and thus the $D flag is specified.

❹ In the left hand column a $% sequence is used to represent a literal percent sign. An unquoted percent sign would be interpreted as a wild card which matches a single character as in the first three entries.

See Section 3.4 for instructions on how to create and use a FORWARD mapping; see Chapter 5 for complete details on the syntax and rules which apply to entries in the FORWARD mapping table.

Example 3–8 Using a FORWARD Mapping Table to Forward Messages

```
FORWARD
  %*.*@example.com              $2$0@example.com$Y ❶
  %*_*@example.com              $2$0@example.com$Y
  "%*$ *"@example.com           $2$0@example.com$Y ❷
!
  *.*@MR.example.com            "$1,$ $0"%A1@MR.VAXA.example.com$Y$D ❸
  *_*@MR.example.com            "$1,$ $0"%A1@MR.VAXA.example.com$Y$D
  "*$ *"@MR.example.com         "$1,$ $0"%A1@MR.VAXA.example.com$Y$D
!
  *-LIST@Obsolete.example.com   $0-L@Listserv.example.com$Y$D
  *$%vax*@VAXA.example.com      $0@VAX$1.example.com$Y$D ❹
```

3.5.2.3 Using the Forward Database to Forward Mail

The forward database can be used to perform forwarding similar to that performed using the alias file or alias database; see Section 3.5.1 above. But when the alias file or alias database can be used, their use is generally preferable to using the forward database as their use is more efficient.

The sort of case where use of the forward database for forwarding mail is appropriate is generally when different sorts of forwarding need to be performed depending upon the source of the message being forwarded. Forward database forwarding can be made source specific, via the USE_FORWARD_DATABASE option. For instance, autoregistered addresses are often a case where source specific forwarding is appropriate; see Section 3.7.

3.6 Centralized Naming

Centralized naming is used when you want to have all incoming messages use a uniform, centralized addressing format (*e.g.,* `John.Doe@example.com`) which you will convert to internal addresses, and for outbound mail have all instances of internal addresses converted to the uniform, centralized format.[b] The advantages to centralized naming include enhanced security as the outside world does not see actual host names or user names, stability in that mail addresses do not become outdated as internal host names change or employees and students change accounts, and ease of use for users in the outside world when the centralized naming scheme replaces possibly awkward internal mailbox names with more presentable and mnemonic mailbox names.

3.6.1 Address Formats for Centralized Names

Before implementing a centralized naming scheme you must first determine what address format you want to use. *First.Last@domain* is becoming more and more popular although there remains the issue of what to do when you have two users named Mike Smith.[c]

Without a doubt, RFC 822 is the most heavily used addressing format in the world today. When choosing a format, make sure that it does not rub RFC 822 the wrong way: even if your organization does not use RFC 822 based messaging, much of the world does. A lot of people outside your organization may want or need to exchange mail with your users. Similarly, your users might find it desirable to have addresses which are easily gatewayed to or through the Internet. If this is not the case, then the likelihood that their addresses will be altered to the point of being unreplyable is increased. Watch out for characters called "specials" in RFC 822 as they will require quoting if they appear in the mailbox part of an address. The RFC 822 specials are

```
SPACE  (  )  <  >  @  ,  ;  :  \  "  .  [  ]
```

Use of any special (other than period, which is handled differently — periods are usually safe as long as they do not appear adjacent to one another or at the very beginning or very end of an address) is disasterous as many misconfigured or otherwise broken mailers do not properly handle addresses with quoting. Underscores, although not a special, can also lead to problems. Many gateways convert spaces to underscores and underscores to something else (hopefully). Often the address is not properly inverted when coming back out the gateway or when it is transported elsewhere. Some messaging formats such as X.400 do not even allow underscores in addresses. As of this writing a heavily used AT&T gateway removes the first underscore and everything following it in the mailbox part of addresses presented to it.

[b] Note that although a single, "central" host name is used by the outside world, this does not mean that all mail from the outside world must flow through a single, central host. For instance, on the Internet, DNS MX records may be used to equate several hosts with the central host name. Preferences may be associated with each host so equated.

[c] The popularity of *First.Last@domain* derives not so much from beauty or elegance, but rather from the fact that it works and interoperates well between many different messaging systems.

3.6.2 Routing Issues in Centralized Naming

By definition, centralized naming is only useful if it is consistently used. This in turn means that the flow of messages within the organization may need to be altered or controlled to insure that the proper name translations are applied at the proper times.

In many cases this happens automatically. For example, suppose PMDF is serving both as a centralized naming authority and as the only gateway between the Internet, a `cc:Mail` postoffice and an ALL-IN-1 installation. In this situation any message travelling between systems has to flow through PMDF and name translations will always be applied.

However, things are not always so simple. Suppose that PMDF is acting as a centralized naming authority for several hundred UNIX workstations, all using SMTP based mail and all with direct Internet access. In this situation messages are likely to escape to the Internet without passing through PMDF and hence without having proper name translations done.

There are two ways to solve this problem. One is to adjust message routing to force all messages to pass through the central naming authority. The other is to distribute the centralized naming functionality, in effect making all the systems into central naming authorities. A combination approach is also possible, where a number of systems act as centralized naming authorities and other systems route mail through an appropriate authority.

3.6.3 Implementing Centralized Names

Once you have settled on a naming scheme and dealt with routing issues it is time to implement it. First you should establish an appropriate forwarding mechanism to convert incoming addresses to your internal format, *i.e.*, to accept incoming mail and forward it to the appropriate internal mailbox. Several methods of doing this are discussed in Section 3.5. A specific example is also presented below. After establishing a method to handle incoming mail, you are ready to begin emitting mail to the outside world using your new address format. This is accomplished with either an address reversal database or `REVERSE` mapping as described in Section 3.3.2. Either of these will convert addresses in outbound mail messages to your centralized naming format. If your format leaves the mailbox portion of addresses alone, then you can accomplish this step more efficiently with domain rewrite rules similar to those described in Section 3.5.2.1. In this case, however, you would use the `$R` flag instead of `$F`.

If the conversion from internal addresses to centralized addresses is very algorithmic and an address in one format contains all of the information required to construct the equivalent address in the other format, then you should be able to use `FORWARD` and `REVERSE` mappings. However, this is rarely the case. At best, it is usually only possible to do this in one direction (e.g., `Jane.Doe@Example.com` → `jdoe@Example.com`). It is far more likely that you will need to use the alias and address reversal databases, possibly in conjunction with the `REVERSE` mapping too. This is the situation covered in the example which follows.

Suppose the domain Example.Com has two general purpose computing machines, `Marvel.Example.Com` and `Admin.Example.Com`, an enclave of cc:Mail users reached via the gateway `ccMail.Example.Com`, and ALL-IN-1 IOS users reached via PMDF-MR using the gateway `A1.Example.Com`. From the outside world all mail to Example employees is addressed to *`first.last`*`@Example.Com` and `Example.Com` is actually the host Marvel.Example.Com. Suppose further that within `ALL-IN-1`, the mailbox name space is of the form "*`Last, First`*", in `cc:Mail` it is "*`First Last`*", and on vaxa.Example.Com and `Admin.Example.Com` mailbox names correspond to usernames. The name space for the four machines is shown in Table 3–6.

Table 3–6 Name Space and Centralized Naming Scheme for Example.Com

Internal address	Centralized address
Marvel.Example.Com mailboxes	
richardsr@Marvel.Example.Com	Mr.Fantastic@Example.Com
grimmb	The.Thing
storms	Invisible.Girl
stormj	Human.Torch
Admin.Example.Com mailboxes	
wchristopher@Admin.Example.Com	Warren.Christopher@Example.Com
lbentsen	Lloyd.Bentsen
laspin	Les.Aspin
jreno	Janet.Reno
bbabbitt	Bruce.Babbitt
mespy	Mike.Espy
rbrown	Ronald.Brown
rreich	Robert.Reich
dshalala	Donna.Shalala
hcisneros	Henry.Cisneros
fpena	Federico.Pena
holeary	Hazel.OLeary
rriley	Richard.Riley
jbrown	Jesse.Brown
ccMail.Example.Com mailboxes	
"Harry Blackmun"@ccMail.Example.Com	Harry.Blackmun@Example.Com
"William Rehnquist"	William.Rehnquist
"John Paul Stevens"	John.Paul.Stevens
"Sandra Day OConnor"	Sandra.Day.OConnor
"Antonin Scalia"	Antonin.Scalia
"Anthony Kennedy"	Anthony.Kennedy
"David Souter"	David.Souter
"Clarence Thomas"	Clarence.Thomas
"Ruth Bader Ginsberg"	Ruth.Bader.Ginsburg

A1.Example.Com mailboxes	
"Burford, Anne"@A1.Example.Com	Anne.Burford@Example.Com
"Deaver, Michael"	Michael.Deaver
"Donovan, Raymond"	Raymond.Donovan
"Meese, Ed"	Ed.Meese
"Nofziger, Lyn"	Lyn.Nofziger

For this case, we can easily map back and forth between the centralized format and `cc:Mail` or `ALL-IN-1` addresses. However, we can only take advantage of this with outbound messages. For inbound messages you still need to know which host, `ccMail.Example.Com` or `A1.Example.Com`, to which to direct a message. We will use an alias database to handle incoming mail.

There should be a `DNS MX` record for `Example.Com` which points to `Marvel.Example.Com`. Moreover, `Marvel.Example.Com` should either use `Example.Com` as its official local host name (host name on the local channel), or rewrite `Example.Com` to its official local host name. By doing this, mail to *user*`@Example.Com` is equated to the local channel and *user* is then looked up in to the alias file or database. This then allows us to use the alias database as a means of forwarding incoming mail for `Example.Com` to its correct, internal destination. Also, `USE_REVERSE_DATABASE` should be set to 5,

```
USE_REVERSE_DATABASE=5
```

in the PMDF option file. A setting of 5 causes PMDF to apply the `REVERSE` mapping and address reversal database to all header addresses instead of just reverse pointing addresses.

The `alias` database source file shown in Example 3–9, the address reversal database source file shown in Example 3–10, and the `REVERSE` mapping table shown in Example 3–11 together implement this centralized naming scheme.[d] For instance, a message coming from `jreno@Admin.Example.Com` will have its `From:` address changed to `Janet.Reno@Example.Com` in accord with the entry in the address reversal database. Similarly messages from `"Harry Blackmun"@ccMail.Example.Com` and `"Sandra Day OConnor"@Example.Com` will have their `From:` addresses changed to `Harry.Blackmun@Example.Com` and `Sandra.Day.OConnor@Example.Com` by the second and first entries in the `REVERSE` mapping table. Incoming mail messages to `Janet.Reno@Example.Com` will be forwarded to `jreno@Admin.Example.Com` in accord with the entry in the `alias` database.

[d] Many sites maintain a single source file from which, using a site-supplied procedure, they generate `alias` and `reverse` database source files.

Example 3–9 Alias Database Source File

```
!
! Marvel.Example.Com users
!
Mr.Fantastic          richardsr@Marvel.Example.Com
The.Thing             grimmb@Marvel.Example.Com
Invisible.Girl        storms@Marvel.Example.Com
Human.Torch           stormj@Marvel.Example.Com
!
! Admin.Example.Com users
!
Warren.Christopher    wchristopher@Admin.Example.Com
Lloyd.Bentsen         lbentsen@Admin.Example.Com
Les.Aspin             laspin@Admin.Example.Com
Janet.Reno            jreno@Admin.Example.Com
Bruce.Babbitt         bbabbitt@Admin.Example.Com
Mike.Espy             mespy@Admin.Example.Com
Ronald.Brown          rbrown@Admin.Example.Com
Robert.Reich          rreich@Admin.Example.Com
Donna.Shalala         dshalala@Admin.Example.Com
Henry.Cisneros        hcisneros@Admin.Example.Com
Federico.Pena         fpena@Admin.Example.Com
Hazel.OLeary          holeary@Admin.Example.Com
Richard.Riley         rriley@Admin.Example.Com
Jesse.Brown           jbrown@Admin.Example.Com
!
! ccMail.Example.Com users
!
Harry.Blackmun        "Harry Blackmun"@ccMail.Example.Com
William.Rehnquist     "William Rehnquist"@ccMail.Example.Com
John.Paul.Stevens     "John Paul Stevens"@ccMail.Example.Com
Sandra.Day.OConnor    "Sandra Day OConnor"@ccMail.Example.Com
Antonin.Scalia        "Antonin Scalia"@ccMail.Example.Com
Anthony.Kennedy       "Anthony Kennedy"@ccMail.Example.Com
David.Souter          "David Souter"@ccMail.Example.Com
Clarence.Thomas       "Clarence Thomas"@ccMail.Example.Com
Ruth.Bader.Ginsburg   "Ruth Bader Ginsberg"@ccMail.Example.Com
!
! A1.Example.Com users
!
Anne.Burford          "Burford, Anne"@A1.Example.Com
Michael.Deaver        "Deaver, Michael"@A1.Example.Com
Raymond.Donovan       "Donovan, Raymond"@A1.Example.Com
Ed.Meese              "Meese, Ed"@A1.Example.Com
Lyn.Nofziger          "Nofziger, Lyn"@A1.Example.Com
```

The alias and address reversal databases are generated from your source files with the `PMDF CRDB` (OpenVMS) or `pmdf crdb` (UNIX or NT) utility as described in Sections 3.1.2 and 3.3.2. See Chapter 29 and Chapter 30 for information on the CRDB or `crdb` utility itself. When generating these databases on VMS, it is best to use an intermediate file so as to eliminate any windows during which the "live" databases are in a mixed state. For instance, use the OpenVMS commands,

Example 3–10 Reverse Database Source File

```
!
! Marvel.Example.Com users
!
richardsr@Marvel.Example.Com     Mr.Fantastic@Example.Com
grimmb@Marvel.Example.Com        The.Thing@Example.Com
storms@Marvel.Example.Com        Invisible.Girl@Example.Com
stormj@Marvel.Example.Com        Human.Torch@Example.Com
!
! Admin.Example.Com users
!
wchristopher@Admin.Example.Com   Warren.Christopher@Example.Com
lbentsen@Admin.Example.Com       Lloyd.Bentsen@Example.Com
laspin@Admin.Example.Com         Les.Aspin@Example.Com
jreno@Admin.Example.Com          Janet.Reno@Example.Com
bbabbitt@Admin.Example.Com       Bruce.Babbitt@Example.Com
mespy@Admin.Example.Com          Mike.Espy@Example.Com
rbrown@Admin.Example.Com         Ronald.Brown@Example.Com
rreich@Admin.Example.Com         Robert.Reich@Example.Com
dshalala@Admin.Example.Com       Donna.Shalala@Example.Com
hcisneros@Admin.Example.Com      Henry.Cisneros@Example.Com
fpena@Admin.Example.Com          Federico.Pena@Example.Com
holeary@Admin.Example.Com        Hazel.OLeary@Example.Com
rriley@Admin.Example.Com         Richard.Riley@Example.Com
jbrown@Admin.Example.Com         Jesse.Brown@Example.Com
```

Example 3–11 REVERSE Mapping Table

```
REVERSE
  "*$ *$ *"@ccMail.Example.Com  $0.$1.$2@Example.Com$Y
  "*$ *"@ccMail.Example.Com     $0.$1@Example.Com$Y
  "*,$ *"@A1.Example.Com        $1.$0@Example.Com$Y
```

```
$ PMDF CRDB alias-source-file PMDF_TABLE:aliases.tmp
$ PMDF CRDB reverse-source-file PMDF_TABLE:reverse.tmp
$ RENAME PMDF_TABLE:aliases.tmp PMDF_ALIAS_DATABASE
$ RENAME PMDF_TABLE:reverse.tmp PMDF_REVERSE_DATABASE
```

or, on UNIX systems, the commands,

```
# pmdf crdb alias-source-file PMDF_ALIAS_DATABASE
# pmdf crdb reverse-source-file PMDF_REVERSE_DATABASE
```

or, on NT systems, the commands,

```
C:\ pmdf crdb alias-source-file PMDF_ALIAS_DATABASE
C:\ pmdf crdb reverse-source-file PMDF_REVERSE_DATABASE
```

If you are using the PMDF multithreaded SMTP server (`tcp_*` channels), then be sure to restart the SMTP server after generating new database files. If you have merely added entries to an existing database file, then you do not need to restart the SMTP server.

3.7 Autoregistration

Some mail systems, such as SNADS mail systems, cannot handle real RFC 822 (Internet) addresses. Typically, one must set up algorithms or table lookups to translate between the real RFC 822 addresses, and addresses that can be used on the SNADS side. So that one does not need to register all possible addresses (the entire Internet) in advance, when sending to such mail systems, it may be useful to *autoregister* addresses; that is, when a previously unseen address appears in a message, automatically generate a translation address for it.

Autoregistration is enabled using the forward database, reverse database, and the special flags `$A` and `$F` in the REVERSE mapping table.

Typically, the autoregistration is only to be performed for messages going out special channels such as SNADS channels, therefore typically a channel specific reverse database and channel specific REVERSE mapping table should be used. If a non-channel specific reverse database was already in use, note that it is not usually necessary to make the entire reverse database channel specific. Instead, it is usually possible to continue using the former non-channel specific entries in the reverse database, as well as the additional new channel specific entries; see below.

The components of setting up autoregistration of addresses are as follows.

1. *Enable use of the forward database.* Set bit 0 (value 1) for the PMDF option USE_FORWARD_DATABASE. A general discussion of PMDF options and the PMDF option file, `option.dat`, may be found in Chapter 7.
2. *Ensure that a forward database exists.* If you do not already have a forward database, then you must create one with the PMDF CRDB (OpenVMS) or `pmdf crdb` (UNIX or NT) utility. *e.g.*, on OpenVMS:

```
$ PMDF CRDB/LONG_RECORDS NLA0: PMDF_FORWARD_DATABASE
```

   or on UNIX:

```
# pmdf crdb -long_records /dev/null PMDF_FORWARD_DATABASE
# chown pmdf /pmdf/table/forwarddb.*
```

   or on NT:

```
C:\ pmdf crdb -long_records nul PMDF_FORWARD_DATABASE
```

3. *Ensure that a reverse database exists.* If you do not already have a reverse database, then you must create one with the PMDF CRDB (OpenVMS) or `pmdf crdb` (UNIX or NT) utility. *e.g.*, on OpenVMS:

```
$ PMDF CRDB/LONG_RECORDS NLA0: PMDF_REVERSE_DATABASE
```

   or on UNIX:

```
# pmdf crdb -long_records /dev/null PMDF_REVERSE_DATABASE
# chown pmdf /pmdf/table/reversedb.*
```

   or on NT:

```
C:\ pmdf crdb -long_records nul PMDF_REVERSE_DATABASE
```

4. *If a new pseudo host or domain name will be used for autoregistration purposes, ensure that the other mail system knows to route that pseudo host name to the PMDF system.* SNADS systems need routing information for any new pseudo host name you will be using in the autoregistration.

5. *Create a PMDF sequence number file for use in the autoregistration.* The typical way to generate unique usernames for autoregistration is to use a PMDF sequence number file. In order to use a PMDF sequence number file for such a purpose, you must first create one; see Section 5.3.2.6 for additional details. *e.g.*, on OpenVMS:

```
$ CREATE/FDL=PMDF_COM:sequence_number.fdl PMDF_TABLE:autoreg.dat
```

Or on UNIX:

```
# su pmdf
% touch /pmdf/table/autoreg.dat
```

Or on NT:

```
C:\ copy nul C:\pmdf\table\autoreg.dat
```

If you prefer to use some other mechanism, say provide your own image which the REVERSE mapping will invoke, that is another option, in which case you will not need such a sequence number file.

6. *Put the entry or entries triggering autoregistration into the REVERSE mapping table.* If you do not already have a PMDF mappings file, then you must create one and to it add a REVERSE mapping table with entries for performing the address autoregistration. (A general discussion of the PMDF mappings file may be found in Chapter 5; a discussion of the REVERSE mapping in particular may be found in Section 3.3.2.)

   There will be two forms of entries, the first form blocking autoregistration of any addresses which should not be autoregistered (such as native SNADS addresses), and the second form performing the autoregistration of any other addresses. Each entry blocking autoregistration will have the general form

   *|*dest-channel*|*orig-address* $N

   while each autoregistration entry will have the general form

   *|*dest-channel*|*orig-address* YA$F*replace-address*

   where *replace-address* is an address with which to replace the real RFC 822 address. For instance, when sending to SNADS, which is limited to an eight character username and an eight character pseudo host name, a typical sort of approach might be to generate eight character pseudo usernames using a PMDF sequence number file, and use a fixed pseudo host name, say INTERNET. For instance, assuming that SNADS hosts are SNADS1, SNADS2, *etc.*, then on OpenVMS:

```
REVERSE
  *|snads_*|*@SNADS1    $N
  *|snads_*|*@SNADS2    $N
! ...etc...
  *|snads_*|*           $Y$A$F$#PMDF_TABLE:autoreg.dat|16|8#@INTERNET
```

or on UNIX:

```
REVERSE

  *|snads_*|*@SNADS1    $N
  *|snads_*|*@SNADS2    $N
! ...etc...
  *|snads_*|*           $Y$A$F$#/pmdf/table/autoreg.dat|16|8#@INTERNET
```

or on NT:

```
REVERSE

  *|snads_*|*@SNADS1    $N
  *|snads_*|*@SNADS2    $N
! ...etc...
  *|snads_*|*           $Y$A$F$#C:\pmdf\table\autoreg.dat|16|8#@INTERNET
```

7. If you want to continue to do non-channel specific reverse database reversals for other addresses, change the `$N` blocking entries shown above to use `$Y$D` to cause the addresses to be passed back for a reverse database lookup, plus add a final REVERSE entry to cause any other addresses to be passed back for a reverse database lookup. For instance, on UNIX:

```
REVERSE

  *|snads_*|*@SNADS1    $Y$D$2@SNADS1
  *|snads_*|*@SNADS2    $Y$D$2@SNADS2
! ...etc...
  *|snads_*|*           $Y$A$F$#/pmdf/table/autoreg.dat|16|8#@INTERNET
  *|*|*                 $Y$D$2
```

Note that these `$Y$D` entries are passing back only the original address for the reverse database probe, and omitting the channel specific information.

8. *Make the reverse database and REVERSE mapping lookups channel specific.* Set the PMDF option USE_REVERSE_DATABASE=29.

9. *Recompile, if necessary, and restart PMDF components.* If using a compiled PMDF configuration, recompile (and on OpenVMS reinstall) it. Restart any resident process components of PMDF, *e.g.*, the SMTP server and the SNADS and PROFS processes.

4 Mailing Lists and MAILSERV

PMDF has flexible mailing list facilities, and facilities for performing automated processing of certain (typically mailing list related) messages.

PMDF mailing lists are implemented as a special form of PMDF alias; controlling mailing list headers, access to post to mailing lists, setting up moderated mailing lists, *etc.*, are controlled via the alias that defines the mailing list. See Chapter 3 for background general information on aliases and on the PMDF alias file and the optional alias database. Details specifically on mailing list definitions, with their many options and variations, are described below in Section 4.1.

PMDF has a facility for providing automated processing of certain sorts of messages, for instance, requests to send particular files, or requests to subscribe or unsubscribe from particular mailing lists. This facility is called MAILSERV. While often used in conjunction with mailing lists, note that MAILSERV per se has nothing to do with mailing list postings; for defining mailing lists, and enabling and restricting mailing list postings, see instead Section 4.1. Rather, MAILSERV is a general facility for certain sorts of automated message handling; MAILSERV can be used independently of whether mailing lists are in use. MAILSERV is described below in Section 4.3.

4.1 Mailing Lists

A mailing list address is a special address created through the alias file or alias database; see Chapter 3 for general background on aliases, the alias file, and alias database. Associated with each mailing list address is a text file which contains one or more mail addresses, or an LDAP URL that returns one or more mail addresses. Note that an LDAP query URL can return multiple addresses either because the LDAP query matches multiple entries containing a desired attribute(s), or because the LDAP query matches a multivalued attribute of a single entry.) This text file or LDAP URL is sometimes referred to as the mailing list or distribution list. When a message is received by PMDF for the mailing list address, the message is then passed on to each address specified in the mailing list file or LDAP URL. Note that addresses in that file or addresses returned by the LDAP URL can themselves be aliases or mailing list addresses.

A mailing list address *alias* with associated mailing list file *file-spec* is specified in the alias file with an entry of the form

```
alias: <file-spec, named-parameters, error-return-address, \
       reply-to-address, errors-to-address, \
       warnings-to-address, comments
```

A mailing list address *alias* with associated LDAP URL *ldap-url* is specified in the alias file with an entry of the form

```
alias: <ldap-url, named-parameters, error-return-address, \
       reply-to-address, errors-to-address, \
       warnings-to-address, comments
```

Similar definitions can also be made in the alias database, (though of course omitting the colon, as just white space separates the alias from its definition in the alias database).

The parameters following the file specification, *file-spec*, or LDAP URL, *ldap-url*, are optional.

file-spec must be a full file path specification (device, directory, *etc.*). All files included in this fashion should, like the alias file itself, be world readable.[1] Addresses should appear one per line in this file and be in RFC 822 format; the addresses can either be "real" addresses or further aliases (but not of the form "*alias*: <*file-spec*"). Mailing list files can include comment lines as well as references to include files via the include operator, <.

ldap-url must be a standard LDAP URL, with the host and port omitted; (host and port are instead specified via the LDAP_HOST and LDAP_PORT PMDF options; see Section 7.3.2. That is,

```
ldap:///dn[?attributes[?scope[?filter]]]
```

where *dn* is required and is a distinguished name specifying the search base. The optional *attributes*, *scope*, and *filter* portions of the URL further refine what information to return. For a mailing list, the desired *attributes* to specify returning would typically be the `mail` attribute (or some similar attribute). The *scope* can be any of `base` (the default), `one`, or `sub`. And the desired *filter* might be to request the return of all objects that are in Department X or that have, say, the "member-of-list-y" attribute. Certain substitution sequences can be used to construct the LDAP search URL; see Table 3–1 for details.

The action of parameters that can add headers can be modified by the special characters shown in Table 4–1, by appending the special character at the end of the value for the parameter.

Table 4–1 Parameter Action Modifiers

Character	Description
	Insert if not already present; inserts as a Resent- if already present
*	Only insert if not already present
&	Insert if not already present; add to old field if already present
^	Delete any old field present; always insert the new field
\	Delete old field and don't insert a new one

Note: If you are using a compiled configuration, then you must recompile (and on OpenVMS reinstall) the configuration in order for a change to the primary alias file, PMDF_ALIAS_FILE, to take effect. Changes made to mailing list files referenced in the alias file take

[1] For mailing lists set up to use deferred expansion, *e.g.*, via a process channel as in Example 4–2, Example 4–3, and Example 4–4, the mailing list file need not be world readable, but rather need only be accessible by the account running PMDF service jobs — usually the SYSTEM account on OpenVMS or the `pmdf` account on UNIX.

effect immediately; that is, recompiling the PMDF configuration is not necessary in order for changes in mailing list files to take effect.

Note: Mailing lists can be tested with the PMDF TEST/REWRITE/CHECK_EXPANSIONS (OpenVMS) or `pmdf test -rewrite -check_expansions` (UNIX and Windows) utility. See Chapter 29 and Chapter 30 for details.

4.1.1 Named Parameters

The *named-parameters* appearing in

```
alias: <file-spec, named-parameters, error-return-address, \
       reply-to-address, errors-to-address, \
       warnings-to-address, comments
```

are used to specify optional modifiers to the list expansion process. There can be zero or more named parameters, separated by commas, and they must appear before any positional parameters (*e.g.,* *error-return-address*, *reply-to-address*, *etc.*). The general syntax of a named parameter is:

```
[name] value
```

Here *name* is the name of the parameter and *value* is its corresponding value. The square brackets are a mandatory part of the syntax: they do not indicate an optional field.

See Table 4–1 for a description of controls on the effect of named parameters relating to the addition of headers, such as specifying whether a header is to be added only if not originally present, or added unconditionally, and whether the header supplements or substitutes for an originally present header.

The available named parameters are:

AUTH_CHANNEL
CANT_CHANNEL
AUTH_CHANNEL is used to specify a source channel that can submit messages to the mailing list. CANT_CHANNEL is used to specify a source channel that can not submit messages to the mailing list. The argument should be a (possibly wildcarded) channel name. AUTH_CHANNEL also accepts a space-separated list of channel names.

AUTH_LIST
CANT_LIST
USERNAME_AUTH_LIST
USERNAME_CANT_LIST
AUTH_LIST is used to specify a list of addresses that are allowed to post to the mailing list. The *value* item must be either the full file path specification for a world readable file containing the list of addresses allowed to post to the list, or an LDAP URL that returns the list of addresses allowed to post to the list. PMDF will match the envelope From: address against the addresses in the list; if no match occurs, the attempted posting fails and an error is returned to the would be postings originator. USERNAME_AUTH_LIST is analogous to AUTH_LIST, but for (possibly wildcarded) usernames rather than addresses; note that usernames are generally only useful for messages submitted from

the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting PMDF process is running. Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, `*`, character. For instance, to specify that only the user JDOE can submit to a list, whether submitting from the L channel or via SMTP (*e.g.*, from a POP or IMAP client that performs SASL SMTP authentication), the USERNAME_AUTH_LIST file would need to contain the entries:

```
JDOE
$*JDOE
```

where the first entry would match for messages submitted from the L channel and the second entry would match for messages submitted via SMTP AUTH. Note that as asterisk is normally a wildcard character, matching of only the exact literal asterisk character is specified by using the dollar character to quote the asterisk.

CANT_LIST has the opposite effect as AUTH_LIST: it supplies the full file path specification of a world readable file containing a list of addresses, or an LDAP URL returning a list of addresses, specifying which addresses can not post to the list. USERNAME_CANT_LIST is analogous to CANT_LIST, but for (possibly wildcarded) usernames rather than addresses; note that usernames are generally only useful for messages submitted from the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting PMDF process is running.

One common use of this facility is to restrict a list so that only list members can post. This can be done by specifying the same file as both the list file and the AUTH_LIST file. For example, assuming that the list is named test-list and the list file is `PMDF_TABLE:test-list.dis`, the alias file entry would be:

```
test-list: <pmdf_table:test-list.dis, \
             [auth_list] pmdf_table:test-list.dis
```

AUTH_MAPPING
CANT_MAPPING

`AUTH_MAPPING` and `CANT_MAPPING` are similar to `AUTH_LIST` and `CANT_LIST` except that they use mappings rather than explicit files of addresses. The *`value`* item associated with these named parameters is the name of a mapping table to use. By default, the mapping is given the envelope `From:` address as input. The transport and application connection information can be added to the input by specifying the `INCLUDE_CONNECTIONINFO` option in the PMDF option file.

If `AUTH_MAPPING` is used at least one mapping entry must match or the posting is rejected. If an entry does match the resulting string is checked; if it begins with an `F`, `f`, `N`, or `n` the posting is rejected. The mailing list will expand normally if the resulting string begins with any other character.

If `CANT_MAPPING` is used, the posting is accepted if no entry matches. If an entry does match the resulting string is checked; if it begins with a `T`, `t`, `Y`, or `y` the posting is accepted. The posting is rejected if the resulting string begins with any other character.

The most common use of AUTH_MAPPING is to restrict postings to all users of a given (usually local) host. For example, if the local host name is ymir.claremont.edu, the following mailing list definition could be used for the gripes-list:

```
gripes: <pmdf_table:gripes-list.dis, [auth_mapping] x-gripes
```

The corresponding mapping file entries would be:

```
X-GRIPES
    *@ymir.claremont.edu    Y
```

Using a mapping table name beginning X- is recommended, so that this private mapping table name will not collide with a standard Process Software mapping table name.

AUTH_USERNAME
CANT_USERNAME

AUTH_USERNAME is used to specify a username or wildcarded username pattern for an account or accounts allowed to post to the list. Note that this is generally only useful for senders submitting from the L channel or via SMTP SASL; for messages submitted from other sources, the messages are considered to be submitted under the username of the PMDF process that received and enqueued the message, *e.g.*, the account under which the PMDF SMTP server is running. Attempted postings from any other sender will be rejected.

CANT_USERNAME can be used to specify a username or wildcarded username pattern for an account or accounts whose postings should be rejected.

Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, *, character. Also note that the asterisk character is normally a wildcard, and must be quoted with the dollar character in order to be interpreted as a literal asterisk character. For instance, to specify that the only sender who can post to a list is user JDOE who will be submitted solely via SMTP with SMTP AUTH, you would use:

```
[AUTH_USERNAME] $*JDOE
```

Without the dollar sign, specifying just *JDOE would allow postings not only from user JDOE but also from any users AJDOE, BOBJDOE, *etc*.

For specifying more than one username (or wildcarded username pattern), see the USERNAME_AUTH_LIST and USERNAME_CANT_LIST parameters described below.

BLOCKLIMIT
LINELIMIT

The BLOCKLIMIT and LINELIMIT parameters can be used to limit the size of messages that can be posted to the list. The *value* item must be an integer number of blocks for [BLOCKLIMIT], or an integer number of lines for [LINELIMIT]. The number of bytes in a block is specified via the BLOCK_SIZE PMDF option; see Section 7.3.5. The default value is 0, meaning that no limit is imposed on the size of message that can be posted to the list (apart, that is, from any channel or system wide limits).

CONVERSION_TAG

The CONVERSION_TAG named parameter can be used to set a tag which conversion file entries can match upon. The *value* item should be the string to use as the tag. For instance, if a list is defined

```
listname: </pmdf/table/listname.dis, [CONVERSION_TAG] listtag
```

then conversion file entries could include a tag=listtag; clause to match. For instance, if for some mailing list it was desired to convert any text/html parts in posted messages

to text/plain, and if a site had an HTML to TEXT convertor called `htmltotextconvert` and had set up the conversion channel and a `CONVERSIONS` mapping table to apply to list postings, then a conversion file entry could be

```
in-chan=*; out-chan=*; in-type=text; in-subtype=html; tag=listtag;
 out-type=text; out-subtype=plain; parameter-copy-0=*;
 command="htmltotextconvert 'INPUT_FILE' 'OUTPUT_FILE'"
```

DEFERRED

The `DEFERRED` named parameter can be used to add a Deferred-delivery: header line. The value should be a date and time, in ISO 8601 P format.

Note that by default PMDF does not honor Deferred-delivery: headers; see Section 2.3.4.19 for a discussion.

ISO 8601 P format is, *e.g.*,

P*year*Y*month*M*week*W*day*DT*hour*H*minute*M*second*S

where the values *year*, *month*, *etc.*, are integer values specifying an offset (delta) from the current time. The initial P is required; other fields can be omitted, though the T is required if any time values are specified.

DELAY_NOTIFICATIONS
NODELAY_NOTIFICATIONS

The `DELAY_NOTIFICATIONS` named parameter requests that `NOTARY` delay notifications be sent for mailing list postings; the `NODELAY_NOTIFICATIONS` named parameter requests that `NOTARY` delay notifications not be sent for mailing list postings. The *value* specification is currently ignored and should always be `NONE`.

ENVELOPE_FROM

This parameter takes a required value specifying an address to replace the message's original envelope `From:` address. This sets only the envelope From: address, unlike the *error-return-address* positional parameter which also sets an `Errors-to:` address.

EXPIRY

The `EXPIRY` named parameter is used to add an Expiry-Date: header line. The value should be a date and time, in ISO 8601 P format (as described for the `DEFERRED` parameter above). The PMDF periodic return job will return messages whose `Expiry-Date:` has passed.

EXPANDABLE
NONEXPANDABLE

The `EXPANDABLE` named parameter is used to specify that the associated list can be expanded (and hence its contents seen) by various protocols which can attempt such an operation. It does not mean, or imply, that the contents of the list will be expanded into message headers. The *value* specification is currently ignored and should always be `NONE`. The `NONEXPANDABLE` named parameter specifies that the associated list can not be expanded. Again, the `value` specified is currently ignored and should always be `NONE`. EXPANDABLE is the default.

NONEXPANDABLE is useful in blocking the expansion of mailing lists via SMTP's EXPN command.

FILTER

The FILTER parameter can be used to specify a filter file to apply on attempted message postings. The argument must be a URL specifying the path to the filter file to apply.

HEADER_ADDITION

HEADER_ADDITION can be used to specify a file of headers to be added to posted messages. The argument must be a full file specification for the file containing headers to be added.

In particular this facility can be used to add the standard mailing list headers defined in RFC 2369. For instance, a site domain.com that has set up a list named listname, using the MAILSERV channel to manage subscription and unsubscription requests, and with certain list information and archives available at an FTP site, might use a header addition file along the lines of the following:

```
List-Help: <ftp://ftp.domain.com/pub/listname-help.txt> (FTP),
     <mailto:mailserv@domain.com?body=send%20/pub/listname-help.txt>,
     <mailto:mailserv@domain.com?body=help> (MAILSERV Instructions),
     <mailto:listname-owner@domain.com?subject=help> (List Manager)
List-Subscribe:
     <mailto:mailserv@domain.com?body=subscribe%20listname>
List-Unsubscribe:
     <mailto:mailserv@domain.com?body=unsubscribe%20listname>
List-Post: <mailto:listname@domain.com>
List-Owner: <mailto:listname-owner@domain.com?Subject=listname>
List-Archive: <ftp://ftp.domain.com/pub/listname/archive/>,
    <mailto:mailserv@domain.com?body=send%20/pub/listname/archive/*>
```

HEADER_ALIAS
HEADER_EXPANSION

The HEADER_ALIAS named parameter forces the use of the original alias in any original headers constructed using this alias. HEADER_EXPANSION forces the alias to expand into its component addresses in any constructed header lines. The *value* specification is currently ignored and should always be NONE. These named parameters correspond to the expand and no-expand options for entries in personal alias databases. HEADER_ALIAS is the default for entries in the system alias file and database.

Note that these parameters are only valid when headers are originally being constructed, as for instance for messages submitted via the L channel. These parameters are not relevant for incoming messages (such as incoming SMTP messages) for which the headers are already present in one form or another.

HOLD_LIST
NOHOLD_LIST
HOLD_MAPPING
NOHOLD_MAPPING

The HOLD_LIST named parameter can be used to specify a list of originator addresses whose attempts to post to the list should be sidelined as .HELD messages. The NOHOLD_LIST named parameter can be used to specify the list of originator addresses whose postings should not be so sidelined, while all other postings will be sidelined. The *value* must be a full file specification for a file of addresses, or an LDAP URL returning a list of addresses.

The HOLD_MAPPING and NOHOLD_MAPPING named parameters are used analogously, but via mapping tables rather than via lists. The *value* should be the name of a PMDF

mapping table.

By default, the mapping tables are given the envelope `From:` address as input. The transport and application connection information can be added to the input by specifying the `INCLUDE_CONNECTIONINFO` option in the PMDF option file.

IMPORTANCE
PRECEDENCE
PRIORITY
SENSITIVITY
The `IMPORTANCE`, `PRECEDENCE`, `PRIORITY`, and `SENSITIVITY` named parameters are used to generate respective headers; the *`value`* specification is inserted on the respective header line.

KEEP_DELIVERY
KEEP_READ
By default, PMDF strips delivery receipt and read receipt requests from messages posted to mailing lists. The `KEEP_DELIVERY` and `KEEP_READ` named parameters can be used to override this behavior, causing PMDF to retain any delivery receipt or read receipt requests, respectively, on messages posted to the list. The *`value`* specification is currently ignored and should always be `NONE`.

Note that passing receipt requests through to mailing lists is quite dangerous; the default behavior of stripping such requests is *strongly* recommended.

MODERATOR_ADDRESS
MODERATOR_LIST
MODERATOR_MAPPING
USERNAME_MODERATOR_LIST
The `MODERATOR_` named parameters are used to establish a moderated mailing list. All postings to the list not originating from a moderator are sent to the list's moderator. The address of the moderator must be specified with the `MODERATOR_ADDRESS` named parameter. The moderator address determines where moderator mail is sent when someone other than the moderator posts. The value of that named parameter is the moderator's address. For example,

```
test-list: <d1:[bob]test.dis, \
           [MODERATOR_ADDRESS] bob@example.com
```

When there can be multiple moderator addresses (for instance, both robert@a1.example.com and bob@example.com), use MODERATOR_LIST, USERNAME_MODERATOR_LIST, or MODERATOR_MAPPING to specify all addresses from which postings should be passed directly to the list and not sent to the list's moderator. MODERATOR_LIST specifies either the name of a file containing a list of moderator addresses, or an LDAP URL returning a list of moderator addresses. USERNAME_MODERATOR_LIST specifies either the name of a file containing a list of (possibly wildcarded) moderator usernames, or an LDAP URL returning a list of (possibly wildcarded) moderator usernames; note that usernames are generally only useful for messages submitted from the L channel or submitted with SASL authentication via SMTP (SMTP AUTH) since for messages submitted from other sources the username will simply be that of the account under which the submitting PMDF process is running. Note that for messages submitted via SMTP with authentication (SMTP AUTH), the username that authenticated will be prefixed with the asterisk, `*`, character. For instance, to specify that only the user JDOE is the list moderator, whether submitting from the L channel or via SMTP (*e.g.*, from a POP or IMAP client that performs SASL SMTP authentication), the USERNAME_MODERATOR_LIST file would need to contain the entries:

```
JDOE
$*JDOE
```

where the first entry would match for messages submitted from the L channel and the second entry would match for messages submitted via SMTP AUTH. Note that as asterisk is normally a wildcard character, matching of only the exact literal asterisk character is specified by using the dollar character to quote the asterisk.

MODERATOR_MAPPING specifies the name of a mapping table used to verify whether or not an address is a moderator address. By default, the mapping is given the envelope From: address as input. The transport and application connection information can be added to the input by specifying the INCLUDE_CONNECTIONINFO option in the PMDF option file.

If a MODERATOR_LIST or MODERATOR_MAPPING parameter is used, thereby specifying who can post directly to the list, then a MODERATOR_ADDRESS parameter should also be present to specify the address to which to send postings not from any moderator.

The use of the MODERATOR_ADDRESS parameter alone, without the MODERATOR_LIST parameter, is equivalent to using MODERATOR_ADDRESS and a MODERATOR_LIST consisting of just the one moderator address.

ORIGINATOR_REPLY
NOORIGINATOR_REPLY

ORIGINATOR_REPLY is used to control whether or not the originator's address is added to any generated Reply-to: header. The *value* item should be the full file path specification for a world readable file containing the list of addresses that should never be added. (This is usually the mailing list itself.) PMDF will match the envelope From: address against the addresses in the list; if no match occurs, the originator's address will be added to any generated Reply-to: header.

NOORIGINATOR_REPLY specifies that any generated Reply-to: header should contain only explicitly specified addresses. The *value* item is ignored. NOORIGINATOR_REPLY is the default.

PUBLIC
PRIVATE

The PUBLIC named parameter specifies that the associated alias is public and hence can appear in any constructed header lines. The *value* specification is currently ignored and should always be NONE. The PRIVATE named parameter specifies that the alias is private and should appear as an empty group construct in message headers. The *value* specification is optional and if specified is used as the name for the group. Neither PUBLIC nor PRIVATE have any effect if the HEADER_EXPANSION named parameter is also specified. These named parameters correspond to the public and private options for entries in personal alias databases. PUBLIC is the default for entries in the system alias file and database.

Note that these parameters are only valid when headers are originally being constructed, as for instance for messages submitted via the L channel. These parameters are not relevant for incoming messages (such as incoming SMTP messages) for which the headers are already present in one form or another.

RECEIVEDFOR
NORECEIVEDFOR
RECEIVEDFROM
NORECEIVEDFROM

These named parameters control features of what appears in the `Received:` header constructed when expanding the alias, and override normal channel `receivedfor`, `noreceivedfor`, `receivedfrom`, or `noreceivedfrom` keyword settings; see Section 2.3.4.62 for a discussion of the channel keywords. The *value* specification is currently ignored and should always be `NONE`.

REPROCESS

The `REPROCESS` named parameter is used to request deferred expansion of the mailing list, where rather than expanding the mailing list "on line", the message should instead be enqueued to the reprocess channel; the reprocess channel can then performing the mailing list processing in a separate step. The *value* specification is currently ignored and should always be `reprocess`.

Use of this parameter defers much of the processing overhead of handling the message to the later step when the reprocess channel runs, rather than doing the processing as the message is initially accepted. This deferred processing can be especially helpful in cases such as incoming SMTP messages addressed to large mailing lists, where "on line" delays could lead to connection time outs.

Use of this parameter as in:

```
listname:       </pmdf/table/listname.dis, [REPROCESS] reprocess
```

thus provides essentially identical functionality as defining a mailing list in two stages through the reprocess channel to obtain deferred expansion (the mailing list addresses aren't even expanded until the reprocess channel runs) such as:

```
listname:        listname-expand@reprocess
listname-expand: </pmdf/table/listname.dis
```

SEQUENCE_PREFIX
SEQUENCE_SUFFIX
SEQUENCE_STRIP

The `SEQUENCE_PREFIX` and `SEQUENCE_SUFFIX` named parameters request that a sequence number be prepended or appended to the Subject: lines of messages posted to the list. The *value* item gives the full file path specification of a sequence number file. This file is read, incremented, and updated each time a message is posted to the list. The number read from the file is prepended, in the case of `SEQUENCE_PREFIX`, or appended, in the case of `SEQUENCE_SUFFIX`, to the message's Subject: header line. This mechanism provides a way of uniquely sequencing each message posted to a list so that recipients can more easily track postings and determine whether or not they have missed any.

By default, a response to a previously posted message (with a previous sequence number) retains the previous sequence number as well as adding a new sequence number to the subject line; the build up of sequence numbers shows the entire "thread" of the message in question. However, the SEQUENCE_STRIP named parameter can be used to request that only the highest numbered, *i.e.*, most recent, sequence number be retained on the subject line. The *value* item is currently ignored and should always be `NONE`.

Important note: To ensure that sequence numbers are only incremented for successful postings, a `SEQUENCE_PREFIX` or `SEQUENCE_SUFFIX` named parameter should always appear as

the last named parameter; that is, if other named parameters are also being used, the SEQUENCE_* named parameter should appear at the end of the list of named parameters.

Sequence number files are binary files and must have the proper file attributes and access permissions in order to function correctly. PMDF will create an appropriate initial sequence number file automatically if sequence numbers are requested for a mailing list created via an alias in the PMDF_ALIAS_FILE or PMDF_ALIAS_DATABASE.[2] This initial sequence number file will be empty (but have the right file attributes and access permissions); the first sequence number based on this file will be "[1]".

TAG

The TAG named parameter can be used to prefix specified text to the Subject: header of posted messages. The *value* item should be the string to be added. (Note that multiple tags may be specified, separated by a vertical bar.) For instance,

```
schedule-list:  <d1:[adam]schedule-list.dis, [TAG] Schedule posting -- , \
                                        [AUTH_LIST] d1:[adam]schedule-list.dis
```

will cause any postings to the list schedule-list to have a Subject: header that begins "Schedule posting –" followed by whatever the original subject of the posting might have been.

USERNAME

The USERNAME named parameter can be used to set the username that PMDF will consider to "own" these mailing list messages. The pmdf qm utility will allow that username to inspect and bounce messages in the queue resulting from expansion of this mailing list. The value item should be the username of the account to "own" the mailing list postings. On OpenVMS, note that the username specified will be forced to uppercase.

4.1.1.1 Specifying Multiple Access Control Parameters

Note that when specifying multiple sorts of access control parameters, the effect is cumulative (a logical AND operation). For instance, specifying both [CANT_LIST] and [AUTH_LIST] on a list means that only those addresses that are in the [AUTH_LIST] and not in the [CANT_LIST] can post to the list.

Note also that the [AUTH_LIST], [AUTH_MAPPING], [CANT_LIST], and [CANT_MAPPING] parameters provide a separate sort of control from [MODERATOR_LIST] and [MODERATOR_MAPPING]; they do different things and can be used effectively in conjunction. The [AUTH_*] and [CANT_*] parameters control who can post at all; only addresses that make it through those access filters then get checked for the next question, the [MODERATOR_*] access filter, of whether the sender can post directly *vs.* whether their attempted posting is referred to [MODERATOR_ADDRESS].

[2] On OpenVMS, these logical names usually point to the files aliases. and aliases.dat, respectively, in the PMDF_TABLE: directory; on UNIX, these options in the PMDF tailor file usually point to the file aliases and to the file aliasesdb.*, respectively, in the /pmdf/table directory; on NT, these NT Registry entries usually point to the file aliases and to the file aliasesdb.* in the PMDF table directory.

4.1.2 Positional Parameters

With one exception, the positional parameters in a mailing list specification provide alternate addresses to which certain sorts of list related activity should be directed (*e.g.,* an address to which errors should be sent rather than back to the list itself).

The positional parameters are so named for a reason: their position in the comma separated list distinguishes which parameter is being specified. When more than one parameter (positional or otherwise) is specified, they must be separated by commas. If you want to specify a positional parameter but omit other positional parameters which come first, then specify asterisks, `*`, for the omitted parameters. For example, in the following

```
test-list: <pmdf_table:test.dis, *, *, \
           postmaster@example.com
```

the *errors-return-address* and *reply-to-address* parameters have been omitted.

See Table 4–1 for a description of controls on the effect of positional parameters, such as specifying whether a header is to be added only if not originally present, or added unconditionally, and whether the header supplements or substitutes for an originally present header.

Without further ado, the positional parameters are:

error-return-address
error-return-address specifies an address to replace the message's regular envelope `From:` address as well as an address to be inserted into the header as an `Errors-to:` address. This address is optional; if no *error-return-address* is specified no replacements or additions will be made.

Note that the address can be specified with an asterisk as a subaddress, as follows:

```
postmaster+*@example.com
```

With this format, the envelope `To:` address is encoded as a subaddress (replacing the asterisk) within the newly formed envelope `From:` address. This means that every address on the mailing list gets a different envelope `From:` address. For example, let's say that `jane@myvax.example.com` is on the mailing list. Her envelope `From:` address would look like:

```
postmaster+jane+40myvax+2eexample+2ecom@example.com
```

reply-to-address
The *reply-to-address* parameter specifies an address to be inserted into the header on a `Reply-to:` header line.

errors-to-address
The *errors-to-address* parameter specifies an address to be placed on the `Errors-to:` header line, if this address should be different from the *error-return-address* that's used as the envelope `From:` address.

warnings-to-address
The *warnings-to-address* parameter specifies an address to be placed on the `Warnings-to:` header line. This header line is not generated if this address is not specified.

comments

The *comments* parameter specifies a string to be placed in a `Comments:` header line. This header line will add to any Comments: header lines already present in the message being posted to the list.

4.1.3 Basic Mailing List Example

Example 4–2 shows a simple alias file for an OpenVMS host example.com which includes a mailing list named staff; Example 4–3 shows a similar alias file for a UNIX host `example.com`. Mail to the staff list should be addressed to `staff@example.com`.[3] Example 4–1 shows the actual file containing the addresses of each member of the mailing list (including some printer and FAX addresses); this file is referenced in the sample `aliases` files shown in Example 4–2 and Example 4–3.

The use of the indirect stafflist alias directed to the process channel defers the processing so that performance as perceived by the sender is improved. A process channel must be defined for this to work. A process channel is automatically defined by the PMDF configuration utility, as well as rewrite rules that rewrite process to the actual name associated with the process channel, usually `process.`*`localhost`*.

The use of the AUTH_LIST named parameter here restricts the list such that only members of the list can post to it. Two positional parameters, *error-return-address* and *comments*, are used. The *error-return-address* parameter specifies the postmaster's address; the *comments* parameter generates a Comments: header line reading "EXAMPLE Computer Center Staff Mailing List" which will appear in each posting to the list.

Example 4–1 Sample Mailing List File `staff.dis`

```
!
! staff.dis -- employee mailing list
sue@example.com
ralph@example.com
"Karl R. Smith" <karl@example.com>
lisa@other.example.com
!
! Addresses for people who do not have regular e-mail
"/at=Mr. Potato/o=Potato Farm/ms=Mail Stop PF/"@printer1.example.com
"/at=Dr. Pepper/fn=1-909-555-1212/"@text-fax.example.com
```

[3] From within VMS MAIL, use in%"staff@example.com" or simply `in%staff` if sending from the node `example.com` itself. The VMS MAIL command `SET FORWARD/USER` can be used to make the simple address staff recognized (as opposed to `in%staff` or `in%"staff@example.com"`):

```
MAIL> SET FORWARD/USER=STAFF "IN%""STAFF@example.com"""
```

Example 4–2 Sample OpenVMS `aliases` File Defining a Mailing List[4]

```
!
! Standard aliases
postmast:    postmaster
postmaster:  ralph@example.com
root:        system
!
! The staff mailing list; errors are sent to postmaster@example.com
staff:       stafflist@process
stafflist:   <pmdf_table:staff.dis, \
               [auth_list] pmdf_table:staff.dis, \
               postmaster@example.com, *, *, *, \
               EXAMPLE Computer Center Staff Mailing List
```

Example 4–3 Sample UNIX `aliases` File Defining a Mailing List[4]

```
!
! Standard aliases
postmast:    postmaster
postmaster:  ralph@example.com
root:        system
!
! The staff mailing list; errors are sent to postmaster@example.com
staff:       stafflist@process
stafflist:   </pmdf/table/staff.dis, \
               [auth_list] /pmdf/table/staff.dis, \
               postmaster@example.com, *, *, *, \
               EXAMPLE Computer Center Staff Mailing List
```

Example 4–4 Sample NT `aliases` File Defining a Mailing List

```
!
! Standard aliases
postmast:    postmaster
postmaster:  administrator
!
! The staff mailing list; errors are sent to postmaster@example.com
staff:       stafflist@process
stafflist:   <C:\pmdf\table\staff.dis, \
               [auth_list] C:\pmdf\table\staff.dis, \
               postmaster@example.com, *, *, *, \
               EXAMPLE Computer Center Staff Mailing List
```

As noted previously, addresses in the mailing file should be in standard RFC 822 format (*e.g.,* not in the form supplied to VMS MAIL).

[4] This example assumes that you have a process channel and a rewrite rule that rewrites the pseudodomain process to the actual pseudodomain name associated with your process channel. If you are using an old configuration generated prior to PMDF V5.1, you can need to add, in addition to a process channel if you are lacking one, an appropriate rewrite rule rewriting process to the actual name associated with your process channel, or change `stafflist@process` to `stafflist@process.`*`localhost`*.

The command PMDF `TEST/REWRITE/CHECK_EXPANSIONS` (OpenVMS) or `pmdf test -rewrite -check_expansions` (UNIX and Windows) can be used to test the list. For instance, on OpenVMS systems use the command

```
PMDF TEST/REWRITE/CHECK_EXPANSIONS/FROM="sue@example.com" staff
```

Or, on UNIX or Windows systems,

```
pmdf test -rewrite -check_expansions -from="sue@example.com" staff
```

Note the use of the `/FROM` (OpenVMS) or `-from` (UNIX and Windows) qualifier to provide an authorized address (as specified in the [AUTH_LIST] list) to treat as the posting address when attempting the list expansion. (By default such a command would use as the posting address the official return address for the local postmaster as specified by the `RETURN_ADDRESS` option in the PMDF option file; that is, by default PMDF would perform its test as if the local postmaster were attempting to send a message to the list.)

4.1.4 Restrictions on Mailing List Aliases

There are some important restrictions that should be observed when using mailing list aliases:

1. The addresses in the mailing list file should be formatted as pure RFC 822 addresses, *e.g., user@host*. Do not try to use DECnet or other routing conventions that you can get away with in the rewrite rules table. Not only can such things fail, they can not produce a visible error (see the next item). Source routes are the only exotica that are permitted.

2. Certain types of bogus addresses in a list alias will *not* generate a "bad address" return message. Specifically, if, for a given address in the list, the system name is illegal or there is a syntax error in the address specification, then the copy of the message to that address can be silently dropped and no one will be the wiser. If the mailing list file associated with an alias does not exist, then mail to the list itself can be dropped. However, errors in the mailbox part of the address (*e.g.,* "no such user") will be handled correctly.

   System managers should take care to test each list they set up to insure that all the recipient addresses are correct. The PMDF TEST/REWRITE/CHECK_EXPANSIONS (OpenVMS) or `pmdf test -rewrite -check_expansions` (UNIX and Windows) utility provide a way to do this. Lists should be checked periodically and also whenever extensive changes are made.

3. PMDF reads the alias file only as each program using PMDF initializes itself. This means that if you are using a permanently resident server (such as the multi-threaded SMTP server, or PMDF-LAN Lotus Notes channels) you should be sure to stop and restart the server each time a mailing list alias definition is changed in the alias file (or any of the files the alias file includes). (The `pmdf restart` utility provides a simple way to restart any such PMDF detached processes.) On the other hand, mailing list files referenced by the alias file are read and reread as needed, so servers need not be restarted when one of these mailing list files is changed.

4. Each PMDF process sees the alias database existing as of when it first needed to access the database. This means that if you are using a permanently resident server (such as the multi-threaded SMTP server, or PMDF-LAN Lotus Notes channels) you should be sure to stop and restart the server each time a mailing list alias definition's left hand side is changed in the alias database, and each time a mailing list definition's right hand side is changed *if* a new alias database file has been created (but not if an existing alias database file was updated "in place" using the PMDF CRDB/APPEND or `pmdf crdb -append` utility). On the other hand, mailing list files referenced by the alias file are read and reread as needed, so servers need not be restarted when one of these mailing list files is changed.

5. The alias file is always read into memory in its entirety each time PMDF is used. All files included by the primary alias file are also loaded into memory. (Mailing list files are not loaded into memory.) The use of a huge alias file can eat up lots of memory. Liberal use of the mailing list reference operator, <, to reference long lists is recommended. Long lists of addresses coded directly into the alias file or any files it includes should be avoided. Use of an alias database for large numbers of aliases is also recommended.

6. Be sure to observe the length restrictions associated with aliases when defining mailing lists, particularly as mailing list definitions can get rather long. Aliases in the alias database can contain up to 32 characters in a short database, up to 80 characters in a long database, and up to 252 characters in a huge database. In the cases of a short database the translation value can contain up to 80 characters; in the case of a long database the translation value can contain up to 256 characters; in the case of a huge database the translation value can contain up to 1024 characters. In some cases failing to observe length restrictions can lead to addresses being silently dropped from lists. Aliases in the alias file can contain up to 60 characters (referring here to the left hand side of the definition). The right hand side of an alias file definition is not specifically limited; however, each physical line is limited to 1024 characters—use the backslash line continuation character to continue a long definition over multiple physical lines. Thus note that particularly long mailing list definitions (definitions involving quite a few of the optional parameters) can, for reasons of length, need to be stored in a huge records alias database or in the alias file, rather than an alias database of shorter records.

4.2 Personal Mailing Lists (OpenVMS and UNIX)

On OpenVMS and UNIX, users can also create their own PMDF personal mailing lists. Such personal mailing lists are defined using the PMDF `DB` utility or PMDF MAIL's `ALIAS` commands on OpenVMS, or using the `pmdf db` utility on UNIX, and the definitions are stored in the user's personal alias database. Such personal mailing list definitions are therefore quite akin to system mailing list definitions that might be stored in the PMDF system alias database, but made at a user level.

Users can either choose to keep their personal mailing list definitions entirely private—only the user himself can use the list definition to send to the list. Or users can choose to mark some of their personal mailing lists as `PUBLIC` lists; in that case, other users can send to the first user's personal mailing list by sending to *`firstuser+listname@domainname`*. That is, users can make PUBLIC (*i.e.*, "publish") their personal list definitions as subaddresses of their own address.

See the appropriate edition of the *PMDF User's Guide* for additional details.

4.3 Mail and List Server

The mail server channel provides a group of automated server facilities that respond to commands sent as electronic mail messages. The basic idea is very simple. A user sends a mail message to the server requesting some type of service. The server responds by sending one or more response messages. The server can also update various local databases as part of the processing of some commands (*e.g.,* mailing lists).

The mail server channel program presently provides three general types of services: distribution of general information about itself, file serving, and mailing list manipulation.

Distribution of general information means that the mail server will send general information about the server, in the form of mail messages, in response to requests for information. These requests are currently limited to the commands `HELP`, `INDEX`, `INFO` (a synonym for `HELP`), and `LISTS`.

File serving means that the mail server will process commands requesting files and will mail the requested files back to the requestor. The `SEND` command is used to request files and the `DIRECTORY` command can be used to find out what files are actually available. The `ENCODING` and `MODE` commands are used to specify how the file or files are read and encoded prior to being sent.

Mailing list manipulation means that the mail server will accept requests for subscription to or removal from various mailing lists that have been placed under the control of the mail server. The `SUBSCRIBE` and `UNSUBSCRIBE` commands perform the basic mailing list functions. The `DIRECTORY/LIST` command can be used to find out what lists are available, and the `SEND/LIST` command can be used to obtain a list of the current subscribers to a given list.[5]

It is important to keep in mind that `MAILSERV` merely handles files; in particular, `MAILSERV` can manage mailing list membership files. But `MAILSERV` has nothing to do with mailing list postings; postings to the list are handled by PMDF proper and not `MAILSERV`. `MAILSERV` merely manages subscriptions to and from the list and queries for a copy of the list. For a complete discussion on PMDF mailing lists, see Section 4.1.

[5] By default, the `SEND/LIST` command can not be used. Access to that command is controlled through the `MAILSERV_ACCESS` mapping table; see Section 4.3.7.

4.3.1 Mail Server Implementation

The mail server is a PMDF channel program associated with the mailserv channel. The mail server runs as a master channel program; there is no slave program associated with the channel.

The mail server only processes mail. It does not process interactive messages as the LISTSERV servers on BITNET do. It is limited to processing mail messages, interpreting the body of the message as commands. *The mail server does not allow access to files outside of the directory tree pointed at by the* PMDF_MAILSERV_FILES_DIR *logical (on OpenVMS systems) or PMDF tailor file option (on UNIX systems) or Registry entry (on NT systems).*

4.3.2 Mail Server Installation and Configuration

The following sections describe the steps necessary to set up the mail server.

4.3.2.1 Setting Up the Channel

The first step to take in activating the mail server is to add an appropriate channel entry to your PMDF configuration file. The channel table entry should have the form:

```
mailserv logging
MAILSERV-DAEMON
```

The logging keyword can be removed if you don't care about logging usage of your mail server.

A couple of alias entries also need to be added to the PMDF alias file. They should look like this:

```
mailserv: mailserv@MAILSERV-DAEMON
mailserv-reply: postmaster
```

The first alias will route mail sent to the mail server on your local host to the MAILSERV channel. The second alias will route any bounced messages (the return address of MAILSERV replies is mailserv-reply@*local-host-name* unless overridden by the MAILSERV_REPLY channel option) to the Postmaster. This will hopefully prevent any mail loops.

An option file can also be specified, if desired. The name of this file should be mailserv_option stored in the PMDF table directory, hence usually the file
PMDF_TABLE:mailserv_option. on OpenVMS or
/pmdf/table/mailserv_option on UNIX or
C:\pmdf\table\mailserv_option on NT.

The available options are:

COMMAND_LIMIT (integer >= -1)

This option specifies the maximum number of commands allowed in a single message to the MAILSERV channel. Any commands beyond the limit will not be processed. When a value of -1 is specified, no limit will be imposed. This is the default.

LIBERAL (0 or 1)

If this option is set to 1, then MAILSERV is more liberal in what it accepts. It will strip out leading quote characters (>, the greater-than sign), and will continue processing after reading an invalid command, and will strip out stray <mailto:...> strings.

LIST_MAPPING_FLAGS (integer)

This option controls the format of the MAILSERV_LISTS mapping table probe. This option takes a bit-encoded value. The default value is 0, meaning that the mapping probe consists simply of the list name. Bit 0 (value 1), if set, prepends the subscribee address and a vertical bar to the mapping probe. Bit 1 (value 2), if set, prepends the address used to send the request to MAILSERV and a vertical bar to the mapping probe.

MAILSERV_PERSONAL (string <= 252 characters long)

This option specifies the contents of the personal name field used in From: headers of messages generated by the MAILSERV channel. When specifying a personal name field to use, there is no need to enclose the field in quotes; *e.g.,* it is valid to specify

```
MAILSERV_PERSONAL=Don't fence me in
```

MAILSERV_REPLY (address <= 252 characters long)

By default, MAILSERV will generate a reply address of mailserv-reply@*local-host* in all messages which it generates. (*local-host* is here the official local host name of the system running the MAILSERV channel.) This default can be overridden with the MAILSERV_REPLY option which can be used to specify an alternate address. Note that list-specific defaults can also be established via the MAILSERV_LISTS mapping table; see Section 4.3.5.

MAXBLOCKS (integer >= -1)

Specifies the maximum number of blocks of data in a single message back from the server. Any response requiring more blocks will be broken into multiple messages, no single part exceeding this limit. MIME's message/partial mechanism is used to "fragment" the message into multiple messages. When a value of -1 is specified, no limit is imposed. This is the default.

Note that the size of a "block" is controlled with the PMDF BLOCK_SIZE option presented in Section 7.1. By default, a block is 1024 bytes.

MAXLINES (integer >= -1)

Specifies the maximum number of lines of data in a single message back from the server. Any response requiring more lines will be broken into multiple messages, no single part exceeding this limit. MIME's message/partial mechanism is used to "fragment" the message into multiple messages. When a value of -1 is specied, no limit is imposed. This is the default.

4.3.2.2 Directories, Logical Names, and Basic Files on OpenVMS

The mail server uses two directories: a file directory (with optional subdirectories) and a mailing list directory.

On OpenVMS, the file directory is located by the PMDF_MAILSERV_FILES_DIR logical name. This is normally a rooted logical name that points to the top directory of the set of directories containing the files that the mail server can distribute. A command such as one of the following might be used to define the logical name:

```
$ define/system/exec/translate=conceal pmdf_mailserv_files_dir disk4:[mailserv.files.]
$ define/system/exec/translate=conceal pmdf_mailserv_files_dir disk2:[mailserv.]
$ define/system/exec/translate=conceal pmdf_mailserv_files_dir disk3:
```

Note that the logical name must either be a rooted reference or a reference to an entire device.

PMDF does not define this logical name itself. It should be defined during system startup immediately after the PMDF startup procedure has been run. This definition should not be placed in the PMDF supplied startup procedure, pmdf_startup.com, as that procedure is replaced when PMDF is updated and any local changes made to it will be lost. A convenient place to put these logical definitions is in the optional site-supplied PMDF_COM:pmdf_site_startup.com file which, as discussed in the *PMDF Installation Guide, OpenVMS Edition*, will be executed automatically, if it exists, by the regular PMDF startup procedure.

The file mailserv_help.sample provided in the PMDF table directory should be copied to the PMDF_MAILSERV_FILES_DIR directory and given the name help.txt. This file describes the commands the mail server understands. It can be modified to include site-specific information if desired. There is no imposed structure for the file help.txt.

The file mailserv_index.sample provided in the PMDF table directory should be revised to reflect the files MAILSERV can provide. Once revised, it should be copied to the PMDF_MAILSERV_FILES_DIR directory and given the name index.txt. There is no imposed structure for the file index.txt.

The second directory is where mailing list files are kept. This directory is located with the PMDF_MAILSERV_MAIL_DIR logical name. This logical name refers to a single directory; *it must not be a rooted logical name.* A command such as one of the following might be used to define this logical name:

```
$ define/system/exec pmdf_mailserv_mail_dir disk2:[mailserv.maillist]
$ define/system/exec pmdf_mailserv_mail_dir disk1:[maillists]
$ define/system/exec pmdf_mailserv_mail_dir disk7:[users.maillists]
```

PMDF does not define this logical name itself. It should be defined during system startup immediately after the PMDF startup procedure has been run. The mailing list services of the mail server will be disabled if this logical name is not defined. Again, this definition should not be placed in the startup procedure supplied by PMDF as that procedure is replaced when PMDF is updated and any changes made to it will be lost.

A file lists.txt should be created and placed in the PMDF_MAILSERV_MAIL_DIR directory. This file should list the mailing lists the mail server handles and should give a brief description of each list. There is no imposed structure for the file lists.txt; it is simply an ordinary text file.

At this point the mail server should be usable. Try it out by sending some requests to MAILSERV on the local host and seeing what responses you get.

4.3.2.3 Directories and Basic Files on UNIX

The mail server uses two directories: a file directory (with optional subdirectories) and a mailing list directory.

On UNIX, the PMDF tailor file option PMDF_MAILSERV_FILES_DIR normally points to /pmdf/mailserv/files and the PMDF tailor file option PMDF_MAILSERV_MAIL_DIR normally points to /pmdf/mailserv/mail. (Note that if you change these tailor file options, your changes will be lost next time you upgrade PMDF.)

These two directories should be created and set to be owned by the pmdf account, *e.g.*,

```
# mkdir -m755 /pmdf/mailserv
# chown pmdf /pmdf/mailserv
# mkdir -m755 /pmdf/mailserv/files
# chown pmdf /pmdf/mailserv/files
# mkdir -m755 /pmdf/mailserv/mail
# chown pmdf /pmdf/mailserv/mail
```

The file mailserv_help.sample provided in the PMDF table directory should be copied to the PMDF_MAILSERV_FILES_DIR directory and given the name help.txt. This file describes the commands the mail server understands. It can be modified to include site-specific information if desired. There is no imposed structure for the file help.txt.

The file mailserv_index.sample provided in the PMDF table directory should be revised to reflect the files MAILSERV can provide. Once revised, it should be copied to the PMDF_MAILSERV_FILES_DIR directory and given the name index.txt. There is no imposed structure for the file index.txt.

A file lists.txt should be created and placed in the PMDF_MAILSERV_MAIL_DIR directory. This file should list the mailing lists the mail server handles and should give a brief description of each list. There is no imposed structure for the file lists.txt; it is simply an ordinary text file.

At this point the mail server should be usable. Try it out by sending some requests to MAILSERV on the local host and seeing what responses you get.

4.3.2.4 Directories and Basic Files on NT

The mail server uses two directories: a file directory (with optional subdirectories) and a mailing list directory.

On NT, the Registry entry PMDF_MAILSERV_FILES_DIR normally points to C:\pmdf\mailserv\files\ and the Registry entry PMDF_MAILSERV_MAIL_DIR normally points to C:\pmdf\mailserv\mail\. These two directories are created by the PMDF installation procedure.

The file mailserv_help.sample provided in the PMDF table directory should be copied to the PMDF_MAILSERV_FILES_DIR directory and given the name help.txt. This file describes the commands the mail server understands. It can be modified to include site-specific information if desired. There is no imposed structure for the file help.txt.

The file mailserv_index.sample provided in the PMDF table directory should be revised to reflect the files MAILSERV can provide. Once revised, it should be copied to the PMDF_MAILSERV_FILES_DIR directory and given the name index.txt. There is no imposed structure for the file index.txt.

A file lists.txt should be created and placed in the PMDF_MAILSERV_MAIL_DIR directory. This file should list the mailing lists the mail server handles and should give a brief description of each list. There is no imposed structure for the file lists.txt; it is simply an ordinary text file.

At this point the mail server should be usable. Try it out by sending some requests to MAILSERV on the local host and seeing what responses you get.

4.3.3 Setting Up Mailing Lists

As mentioned previously, it is important to keep in mind that MAILSERV list membership files are solely that—list membership files. Postings to the list are handled by PMDF proper and not MAILSERV, which merely manages subscriptions to and from the list and handles queries for a copy of the list. For a complete discussion of PMDF mailing lists, see Section 4.1.

Each mailing list maintained by the mail server must be set up by creating a world readable[6] mailing list distribution file in the PMDF_MAILSERV_MAIL_DIR directory and adding an entry for the list to the PMDF alias file (or alias database). The mailing list distribution file has the same format as the mailing list distribution files used by the alias file. See Section 4.1. For example, suppose you want to create a new mailing list called info-boink. First, create the (initially empty) distribution list file and set its protections: on OpenVMS,

```
$ create pmdf_mailserv_mail_dir:info-boink.dis -
_$   /protection=(s:rwed,o:rwed,g:re,w:re)
^Z
$
```

or on UNIX,

```
# touch /pmdf/mailserv/mail/info-boink.dis
# chmod 755 /pmdf/mailserv/mail/info-boink.dis
# chown pmdf /pmdf/mailserv/mail/info-boink.dis
```

or on NT,

```
C:\> copy nul C:\pmdf\mailserv\mail\info-boink.dis
```

[6] If the mailing list is set up to use deferred expansion, *e.g.*, via the process channel, then the mailing list file need not be world readable, but rather need only be accessible by the account running PMDF service jobs — usually the SYSTEM account on OpenVMS or the pmdf account on UNIX.

Next the aliases for the list should be added to the aliases file, PMDF_ALIAS_FILE, (or added to the alias database). Large lists should be set up for deferred address expansion. (This can prevent remote mail connections from timing out while the mailing list is expanded.) The following lines would be added to the aliases file for the `info-boink` list on OpenVMS:

```
info-boink: info-boink-expand@process
info-boink-expand: <PMDF_MAILSERV_MAIL_DIR:info-boink.dis, \
                    info-boink-error
info-boink-request: mailserv
info-boink-error: postmaster
```

or on UNIX:

```
info-boink: info-boink-expand@process
info-boink-expand: </pmdf/mailserv/mail/info-boink.dis, \
                    info-boink-error
info-boink-request: mailserv
info-boink-error: postmaster
```

or on NT:

```
info-boink: info-boink-expand@process
info-boink-expand: <C:\pmdf\mailserv\mail\info-boink.dis, \
                    info-boink-error
info-boink-request: mailserv
info-boink-error: postmaster
```

The definition as shown above requires that your PMDF configuration include a process channel; note that the PMDF configuration utility normally creates such a process channel. See Section 26.7 for more details on the process channel.

The `info-boink` mailing list should now be ready to use. These steps should be repeated to set up any additional mailing lists to be maintained by the mail server.

VMS

On OpenVMS, for the convenience of your VMS MAIL users, you can add a VMS MAIL forwarding address so that VMS MAIL users can simply send to `info-boink` (rather than having to send to IN%info-boink):

```
$ MAIL
MAIL> SET FORWARD/USER=INFO-BOINK IN%info-boink
```

UNIX

On UNIX, the case of the mailing list files is important. If the mailing list file is in all lowercase, then users may specify the list name in any mixture of upper and lower case on their MAILSERV commands. However, if the mailing list file is in mixed or all upper case, then users must specify the exact same matching case in their MAILSERV commands.

4.3.4 Welcome Messages for Mailing Lists

It is possible to have a welcome message sent to subscribers when they subscribe to a list with the SUBSCRIBE command. When a user is subscribed to the list *list*, MAILSERV will search for the file *list*.txt in the PMDF_MAILSERV_MAIL_DIR directory. If this file exists, then it will be sent to the new subscribee. Welcome messages generally contain such information as who maintains the list, list usage policies, the list's posting address, *etc.*.

4.3.5 List and File Name Mapping, and From: Address Control

The PMDF mail server provides a facility which can be used to transform the list and file names specified in user commands prior to actual use. For instance, this can be used to provide access to the same underlying list membership file under two different apparent names. It can also be used to allow use of apparent list names containing characters not allowed in OpenVMS file names. The two mappings MAILSERV_FILES and MAILSERV_LISTS provide this functionality.

The MAILSERV_FILES mapping, if present, is applied to all file names. The mapping table template (right hand side) should set either the $Y or $T flag if the string result produced by the mapping should replace the file that the user specified; if neither of these are set then the results of the mapping will be ignored.

The MAILSERV_LISTS mapping, if present, is applied to all list names. The mapping table template (right hand side) should set either the $Y or $T flag if the string result produced by the mapping should replace the file or list name the user specified. If the $A flag is specified, then the string result produced by the mapping will be used as the From: address in any MAILSERV messages sent back to the user regarding this list. If no flags are set, then the results of the mapping will be ignored. Both a replacement list name and a From: address can be specified by separating them with a comma, *e.g.*,

```
MAILSERV_LISTS

  externalname       $Y$Ainternalname,externalname-request@example.com
```

4.3.6 Default List Name Constructed From To: Address

PMDF will use the username of the To: address to which a message was sent as a default list name. That is, if a user submits a command to MAILSERV where that command does not include an explicit list name, the default list name based on the To: address will be used. Configuring to take advantage of this feature is usually done by aliasing addresses of the form *listname*-request@*localhost* to point to *listname*@MAILSERV.

For instance, if an alias such as

```
listname-request:  listname@MAILSERV-DAEMON
```

is created, then a user can send a message to `listname-request` consisting of just the command SUBSCRIBE with no argument in order to subscribe to the list named `listname`.

4.3.7 Access Control

Access to each type of mail server functionality is controlled using the `MAILSERV_ACCESS` mapping table in the PMDF mapping file. Use of this mapping is optional; reasonable defaults are assumed for each sort of access if no mapping is specified.

Access control is necessarily based on addressing information. Since it is in practice possible to forge any sort of address, simple `From:` address information access checks offer only marginal protection at best. Although they can make it difficult for unsophisticated users to unintentionally cause damage they offer no protection at all against malicious attack.

So for greater protection, `MAILSERV` can be configured to generate a challenge-response double-check; see Section 4.3.7.4.

4.3.7.1 Access Check Strings

Each command presented to the mail server is used to compose one or more access query strings. The MAILSERV_ACCESS mapping is then applied to each of these strings. The result of the mapping is examined and determines whether or not the requested operation is allowed. If the operation is not allowed the mail server returns an indication to the requestor indicating that an access failure has occurred.

The access query strings are always in one of the following two formats:

command-keyword|*command-parameter*|*address*
command-keyword|*command-parameter*|*address1*|*address2*

command-keyword is derived from the name of the command being checked. It will be `DIRECTORY` for the DIRECTORY command, `DIRECTLIST` for the DIRECTORY/LIST command, `PURGELIST` for the PURGE/LIST command, `SEND` for the SEND command, `SENDLIST` for the SEND/LIST command, `SUBSCRIBE` for the SUBSCRIBE command, and `UNSUBSCRIBE` for the UNSUBSCRIBE command. Although commands can be abbreviated, abbreviations do not carry over into the *command-keyword* strings.

`ERRORHELP` is a special *command-keyword* used to construct entries specifying an error message file to send back to users in response to any errors processing the users' commands.

The *command-parameter* depends on the command. In the case of the file commands, DIRECTORY and SEND, the parameter is the name of the particular file being accessed. The file name string consists of the directory specification, if any, (that is, the subdirectory, if any, under PMDF_MAILSERV_MAIL_DIR), the file name, and

the file type. Wildcards don't carry over into access strings; the wildcard expansion process is done first and then each resultant file generates a separate access check. In the case of list commands, DIRECTORY/LIST, PURGE/LIST, SEND/LIST, SUBSCRIBE, and UNSUBSCRIBE, the *command-parameter* is the name of the list; more precisely, it is the filename of the list, without the `.dis` extension. Wildcards are once again expanded prior to doing any access checks.

When a mail server request involves only one address, the single-address form of access query string is built, and *address* is the address of the originator (envelope From: address) for the request. In some cases, notably the SUBSCRIBE and UNSUBSCRIBE commands, two addresses can be involved — the address responsible for the request and the address on whose behalf the request is presented, (*i.e.,* the address for which the request is being made). In these cases the two-address form of access query string is used, where the request is made by *address1* for *address2*.

Note that a user's own subaddress is not considered to be a second address in that the one address form of access query string is constructed for the case of a user subscribing or unsubscribing one of their own subaddresses, *i.e.*, a user *user* subscribing or unsubscribing an address of the form *user+string*.

These rules are summarized in Table 4–2.

Table 4–2 Access Check Strings

MAILSERV command	MAILSERV_ACCESS check string format	Default access
DIRECTORY *file-spec*	DIRECTORY|*file-spec*|*from-address*	Allowed
DIRECTORY/LIST *list-name*	DIRECTLIST|*list-name*|*from-address*	Allowed
DIRECTORY/LIST	DIRECTLIST|*list-name*|*from-address*	Allowed
HELP	HELP|HELP.TXT|*from-address*	Allowed
INDEX	INDEX|INDEX.TXT|*from-address*	Allowed
LISTS	LISTS|LISTS.TXT|*from-address*	Allowed
PURGE/LIST *list-name*	PURGELIST|*list-name*|*from-address*	Disallowed
SEND *file-spec*	SEND|*file-spec*|*from-address*	Allowed
SEND/LIST *list-name*	SENDLIST|*list-name*|*from-address*	Disallowed
SET MAIL *list-name*	SETMAIL|*list-name*|*from-address*	Disallowed
SET MAIL *list-name other-address*	SETMAIL|*list-name*|*from-address*|*other-address*	Disallowed
SET NOMAIL *list-name*	SETMAIL|*list-name*|*from-address*	Disallowed
SET NOMAIL *list-name other-address*	SETMAIL|*list-name*|*from-address*|*other-address*	Disallowed
SUBSCRIBE *list-name*	SUBSCRIBE|*list-name*|*from-address*	Allowed
SUBSCRIBE *list-name subaddress*	SUBSCRIBE|*list-name*|*from-address*	Allowed
SUBSCRIBE *list-name other-address*	SUBSCRIBE|*list-name*|*from-address*|*other-address*	Allowed
UNSUBSCRIBE *list-name*	UNSUBSCRIBE|*list-name*|*from-address*	Allowed
UNSUBSCRIBE *list-name subaddress*	UNSUBSCRIBE|*list-name*|*from-address*	Allowed
UNSUBSCRIBE *list-name other-address*	UNSUBSCRIBE|*list-name*|*from-address*|*other-address*	Disallowed
invalid command	ERRORHELP|*from-address*|*from-address*	Allowed

Note: As always for PMDF mapping tables, if using entries that contain wildcards, *e.g.*, `*` or `%`, it is important to put more specific entries before less specific entries. And keep in mind that wildcard matches can include the vertical bar character, `|`; or in other words, a wildcard such as an asterisk can match *across* a vertical bar. In particular, for MAILSERV_ACCESS mapping SUBSCRIBE and UNSUBSCRIBE checks note that one should put two address checks *before* wildcarded one address checks.

4.3.7.2 Access Check Mapping Results

Access check mapping entries set flags to indicate whether or not the request should be honored. A $Y or $T specifies that the request should be honored. A $N or $F indicates that the request should be rejected.

A $< specifies that the entry has returned a file name. The file name should be specified as a full absolute path. This file is opened and read as a series of addresses. The request is rejected if the requestor's address does not appear in the list. A $> does the same thing except that rejection occurs if the requestor's address is in the list. $< and $> cannot be used in the same entry; if they are the result is unpredictable.

For an entry that would otherwise succeed, a $* specifies that the entry has returned a referral address. Instead of honoring the request directly the mail server forwards the request to the specified referral address. The request is rejected if the referral address is bogus. A message is also sent to the requestor indicating that his or her request has been referred elsewhere. This message can be suppressed by appending $S to the mapping's result. The first line in the example below allows user@a.b.c.d to subscribe others to the `info-boink` list; all others who try to subscribe to the list will get referred to user@a.b.c.d.

```
MAILSERV_ACCESS

    SUBSCRIBE|info-boink|user@a.b.c.d|*    $Y
    SUBSCRIBE|info-boink|*                 $*user@a.b.c.d
```

To specify referral for a command that would normally fail, such as by default third party UNSUBSCRIBES, note that one must specify $Y as well as $*, *e.g.*,

```
MAILSERV_ACCESS

    UNSUBSCRIBE|info-boink|user@a.b.c.d|*    $Y
    UNSUBSCRIBE|info-boink|*|*               $Y$*user@a.b.c.d
```

If both $* and $< or $> are used simultaneously the string returned by the mapping entry should consist of the file name, a comma, and then the referral address.

For SUBSCRIBE, UNSUBSCRIBE, and SENDLIST access check mapping entries, a $K specifies that, rather than immediately performing the requested action, MAILSERV should send a message to the address in question (the [un]subscribee address or the address apparently making the SEND/LIST request) asking the user to confirm the requested action. This MAILSERV message will contain a *cookie*—a string that the user must include in a confirmation message, if they want the action to be performed. See Section 4.3.7.4 for further details.

For a SENDLIST access check mapping entry, a $X specifies that by default, any RFC 822 comment fields should be stripped from the distribution list sent back to the user. A user who is allowed to get a copy of the list (note that SEND/LIST is disabled by default) can override this default with the optional /COMMENTS qualifier of the SEND/LIST command.

Normally, a successful SUBSCRIBE or UNSUBSCRIBE causes a "You have been [un]subscribed to/from the list ..." message to be sent to the [un]subscribee address. In a SUBSCRIBE or UNSUBSCRIBE access check mapping entry, a $D alters that behavior. When $D is specified, the usual notification message to the [un]subscribee address in the case of a third party [un]subscribe can be blocked by specifying /NONOTIFY; a notification message back to the [un]subscriber will still be sent.

The available flags are summarized in Table 4–3.

Table 4–3 MAILSERV_ACCESS Mapping Table Flags

Flag	Description
$Y	Honor the request
$T	Honor the request
$N	Do not honor the request
$F	Do not honor the request
$*	If honoring, refer request to the specified address
$K	If honoring, send a cookie response back to the address in question; see Section 4.3.7.4
$S	Suppress "Your request has been referred to..." messages
$D	Honor /[NO]NOTIFY requests on [UN]SUBSCRIBE commands
$<	Honor requests from senders in the specified file
$>	Do not honor requests from senders in the specified file
$V	When performing a third party command where $K is set, send the "please confirm" message to the address issuing the command, rather than the address on whose behalf the command was issued
$X	When checking a SEND/LIST command, default to not including comments in returned list

Flag comparisons	Description
$:K	Match only when processing the confirmation (response to $K) of a command

4.3.7.3 Access Defaults

The DIRECTORY, DIRECTORY/LIST, SEND, and SUBSCRIBE commands all allow full access if no access mapping is provided or the access check string does not match any mapping entry. The SEND/LIST and PURGE/LIST commands refuse all access and the UNSUBSCRIBE command only allows users to unsubscribe themselves from lists and no one else. See Table 4–2 for a summary of these defaults.

4.3.7.4 Access Confirmation via a Challenge-Response Cycle

Given the nature of contemporary messaging protocols, it is fairly easy to forge an e-mail address. Thus the security offered by regular mail server access checks, which are primarily e-mail envelope `From:` address based, is rather fragile; it can protect against naive users, but is not sufficient to protect against a malicious attacker who forges his envelope `From:` address.

Greater security for some commands can be obtained by having `MAILSERV` engage in a challenge-response cycle, by setting the `$K` flag in appropriate `MAILSERV_ACCESS` access check entries.

When a successful `MAILSERV_ACCESS SUBSCRIBE`, `UNSUBSCRIBE`, or `SENDLIST` access check entry returns the `$K` flag, then rather than immediately performing the requested operation, `MAILSERV` instead sends a challenge message to the purported address to be affected by the command. (This would be the purported From: address for most commands, or the subscribee or unsubscribee address for third party `SUBSCRIBE` or `UNSUBSCRIBE` commands.) The challenge message from `MAILSERV` will contain a cookie string—the user will have to confirm the request via a response including that exact cookie string.

For instance, suppose a site example.com wants to allow any users in the example.com domain to subscribe themselves, or any other address, to the list, but does not want to allow any non-example.com addresses to perform any subscriptions to the list. Then a `MAILSERV_ACCESS` mapping such as the following could be used:

```
MAILSERV_ACCESS

  SUBSCRIBE|example-list|*@example.com|*    $Y$K
  SUBSCRIBE|example-list|*|*                $N
  SUBSCRIBE|example-list|*@example.com      $Y$K
  SUBSCRIBE|example-list|*                  $N
```

With such a mapping in effect, suppose a malicious user `forger@somewhere.edu` sends a message to `MAILSERV`. For this message they forge the envelope `From:` address to appear to be `John.Doe@example.com`—an address apparently within `example.com`. Suppose this forged message is a third-party subscribe request— the apparent (forged) `John.Doe@example.com From:` address requesting that the malicious user's true address `forger@somewhere.edu` be subscribed. But because of the `$K` on the first entry in this sample `MAILSERV_ACCESS` table, `MAILSERV` does not directly subscribe the malicious outside user to example-list; instead, `MAILSERV` sends a message to `John.Doe@example.com` including a challenge (cookie) string. Unless `John.Doe@example.com` sends back a confirmation message including the cookie, the subscribe of `forger@somewhere.edu` will not be performed. In particular, `forger@somewhere.edu` would have to guess or otherwise obtain the cookie string in order to achieve their attempted subscribe.

On an implementation note, any initial messages awaiting cookie confirmation are stored in the directory `PMDF_QUEUE:[mailserv.spool]` (OpenVMS) or `/pmdf/queue/mailserv/spool/` (UNIX); if no cookie confirmation is received, such messages will time out after seven days and be returned to the sender (envelope `From:` address).

4.3.7.5 Access Example

The following mapping controls access to the `info-boink` list. It specifies that `user@a.b.c.d` has full access to the list and handles subscription requests by referral. Users in domain `f.g.h.i` cannot access the list in any way. It also specifies that anyone not in the domain `f.g.h.i` can retrieve a copy of the list membership.

```
MAILSERV_ACCESS

    *|info-boink|user@a.b.c.d|*            $Y
    *|info-boink|user@a.b.c.d               $Y
    *|info-boink|*@f.g.h.i|*               $N
    *|info-boink|*@f.g.h.i                 $N
    SUBSCRIBE|info-boink|user@a.b.c.d|*    $Y
    SUBSCRIBE|info-boink|*                 $*user@a.b.c.d
    SENDLIST|info-boink|*                  $Y
```

4.3.8 Server Commands

The commands recognized by the mail server are described in the *PMDF User's Guide*. A brief description of each command is given in Table 4–4.

Table 4–4 Summary of Mail and List Server Commands

Command	Description
CONFIRM	Confirm a command from a previous message
DIRECTORY	List available files
DIRECTORY/LIST	List available mailing lists
ENCODING	Set default file transmission encoding
END	Terminate processing, accept no additional commands
EXIT	Synonymous with END
FINISH	Synonymous with END
HELP	Obtain the site-supplied file containing help information
INDEX	Obtain the site-supplied file containing an index of available files
LISTS	Obtain the site-supplied file describing available mailing lists
MAXIMUM	Set maximum message size; larger messages will be fragmented
MODE	Set the default file access mode
PURGE/LIST	Remove comment lines from the list file
QUIT	Synonymous with END
SEND	Send the specified files
SEND/LIST	Send the membership list for a given mailing list
SEND/LIST/COMMENTS	Send the membership list for a given mailing list, including members' RFC 822 comment fields
SEND/LIST/NOCOMMENTS	Send the membership list for a given mailing list, stripping members' RFC 822 comment fields
SET MAIL	Set an address to receive list postings
SET NOMAIL	Set an address to not receive list postings
STOP	Synonymous with END
SUBSCRIBE	Subscribe to a mailing list
SUBSCRIBE/NOMAIL	Subscribe to a mailing list an address which should not receive list postings

Table 4–4 (Cont.) Summary of Mail and List Server Commands

Command	Description
SUBSCRIBE/NOTIFY	Subscribe to a mailing list, requesting a notification message to the subscribee address in the case of a third-party subscribe
SUBSCRIBE/NONOTIFY	Subscribe to a mailing list, requesting no notification message to the subscribee address in the case of a third-party subscribe
UNSUBSCRIBE	Unsubscribe from a mailing list
UNSUBSCRIBE/NOTIFY	Unsubscribe from a mailing list, requesting a notification message to the unsubscribee address in the case of a third-party unsubscribe
UNSUBSCRIBE/NONOTIFY	Unsubscribe from a mailing list, requesting no notification message to the unsubscribee address in the case of a third-party unsubscribe

4.3.9 MAILSERV Channel Usage Logging

The `logging` channel keyword, described in Section 2.3.4.84, can be used with the `MAILSERV` channel to enable logging activity. Activity is logged in the PMDF mail log file. See Section 31.1 for a general description of this file. When logging is enabled, then in addition to normal channel enqueue and dequeue entries, the `MAILSERV` channel will also write a log entry for each command processed. The "type of entry" item in each such log file entry will be a single character selected from the set `F`, `S`:

Logging Code	Type	Description
F	Failure	Error processing command; command not executed
S	Success	Command successfully executed

The format of the log file entries for the `MAILSERV` channel command processing will be:

1. The date the entry was made (*e.g.,* "29-NOV-2012").
2. The time the entry was made (*e.g.,* "12:13:18").
3. The name of the channel logging the entry ("mailserv").
4. The beginning time and ending time of the execution of the command in HHMMSSHHMMSS format.
5. The success (S) or failure (F) of the command.
6. The "message size" field will always be zero, 0, for `MAILSERV` command entries.
7. The envelope `From:` address of the originator of the message whose command is being executed.
8. The `MAILSERV` command.
9. The number of seconds spent processing the command.

4.4 Examples of Mailing Lists with MAILSERV Subscription Handling

Mailing lists and MAILSERV are separate facilities in PMDF: mailing lists are set up, and postings to them controlled, via definition in the PMDF alias file or alias database, whereas MAILSERV is a general file serving facility that can also handle automated mailing list subscription and membership requests. Mailing lists can be used with or without MAILSERV, and MAILSERV can be used with or without mailing lists.

Nevertheless, it is common task to set up mailing lists whose subscription commands are to be handled by MAILSERV. This section discusses a few sample mailing list scenarios.

4.4.1 An Open, With Exceptions, Mailing List

This section will discuss an example of a mailing list for which no general stringent posting or subscription restrictions are to be imposed, with the exception of a "list owner" maintained subsidiary list of addresses specifically prohibited from posting to the list. Such a list definition in the PMDF alias file might be:

```
open-list: open-list-expand@process
open-list-expand: <PMDF_MAILSERV_MAIL_DIR:open-list.dis, \
                  [CANT_LIST] PMDF_MAILSERV_MAIL_DIR:open-list-reject.dis, \
                  [USERNAME] open-list-owner, \
                  [HEADER_ADDITION] PMDF_TABLE:open-list-headers.txt, \
                  open-list-owner@example.com, \
                  open-list@example.com
open-list-request: MAILSERV
```

In this definition, the only access control established for list postings is that no addresses in the PMDF_MAILSERV_MAIL_DIR:open-list-reject.dis file can post; this file might initially be empty, to be added to by open-list-owner@example.com if abusive postings are received.

The above list definition references a file of headers to be added to messages posted to the list; such a file might be:

```
List-Help: <mailto:mailserv@example.com?body=help> (MAILSERV Instructions),
     <mailto:open-list-owner@example.com?subject=help> (List Manager)
List-Subscribe:
     <mailto:open-list-request@example.com?body=subscribe%20open-list>
List-Unsubscribe:
     <mailto:open-list-request@example.com?body=unsubscribe%20open-list>
List-Post: <mailto:open-list@example.com>
List-Owner: <mailto:open-list-owner@example.com?Subject=open-list>
```

For the list open-list, third party subscribes by anyone other than open-list-owner@example.com will be disallowed, but all other subscribes will be permitted. open-list-owner@example.com will be permitted to perform third party unsubscribes, but all others can only unsubscribe themselves. Members of the list will be allowed to request the list membership file. MAILSERV responses to user messages to MAILSERV regarding the open-list list will have a From: address of open-list-owner@example.com.

The open-list-reject list will also be handled by MAILSERV; only the open-list-owner@example.com address will have any access to this subsidiary list.

```
MAILSERV_ACCESS

  SUBSCRIBE|open-list|open-list-owner@example.com|*          $Y      (1)
  SUBSCRIBE|open-list|open-list-owner@example.com            $Y      (2)
  SUBSCRIBE|open-list|*|*                                    $N      (3)
  SUBSCRIBE|open-list|*                                      $Y      (4)
  UNSUBSCRIBE|open-list|open-list-owner@example.com|*        $Y      (5)
  UNSUBSCRIBE|open-list|open-list-owner@example.com          $K$Y    (6)
  UNSUBSCRIBE|open-list|*|*                                  $N      (7)
  UNSUBSCRIBE|open-list|*                                    $Y      (8)
  SENDLIST|open-list|open-list-owner@example.com             $Y      (9)
  SENDLIST|open-list|*                                       \
                        $X$<PMDF_MAILSERV_MAIL_DIR:open-list.dis     (10)
  PURGELIST|open-list|open-list-owner@example.com            $Y      (11)
!
  SUBSCRIBE|open-list-reject|open-list-owner@example.com|*   $K$V$Y$D (12)
  SUBSCRIBE|open-list-reject|*                               $N      (13)
  UNSUBSCRIBE|open-list-reject|open-list-owner@example.com|* $K$V$Y$D (14)
  UNSUBSCRIBE|open-list-reject|*                             $N      (15)
  SENDLIST|open-list-reject|open-list-owner@example.com      $Y      (16)
  PURGELIST|open-list-reject|open-list-owner@example.com     $Y      (17)
  DIRECTLIST|open-list-reject|*                              $N      (18)
  *|open-list-reject|*                                       $N      (19)

MAILSERV_LISTS

  open-list          $Aopen-list-owner@example.com                   (20)
```

More specifically:

(1) This entry specifies that open-list-owner@example.com can subscribe other addresses to the list.

(2) This entry specifies that open-list-owner@example.com can subscribe himself to the list. Note that (1), above, does *not* enable this; in order to allow open-list-owner@example.com to subscribe himself, this separate entry is required.

(3) This entry disallows third party subscribes: userA cannot subscribe userB to the list. The earlier entry, (1), explicitly allows open-list-owner@example.com to perform third-party subscribes of other addresses; if anyone else attempts to perform a third-party subscribe, the attempt will fall-through to this entry and be denied.

(4) This entry allows general users to subscribe themselves. Note that this is a default behavior and hence this entry is, strictly speaking, redundant. However, it is included for completeness and clarity.

(5) This entry specifies that open-list-owner@example.com can unsubscribe other addresses from the list.

(6) This entry specifies that open-list-owner@example.com can unsubscribe himself, though due to the $K in the entry, MAILSERV will double check and ask him to confirm any such request. Note that (5), above, does *not* enable open-list-owner@example.com to unsubscribe himself; (5) applies only to third party unsubscribes.

(7) This entry disallows third party unsubscribes in general. Note that third party unsubscribes are disallowed by default, so this entry is not, strictly speaking, necessary; however, it is included for completeness and clarity.

❽ This entry explicitly allows general users to unsubscribe themselves; note that such direct unsubscribes are allowed by default, so this entry is not, strictly speaking, necessary; however, it is included for completeness and clarity.

❾ This entry allows `open-list-owner@example.com` to request and receive a copy of the list file. As opposed to the next entry, ❿, note that a `$X` is *not* specified on this entry; comments in the list entries (such as `MAILSERV` notes regarding who subscribed the addressee and when) will be included by default if `open-list-owner@example.com` requests a copy of the list.

❿ This entry allows anyone already subscribed to the list to request and receive a copy of the list file. The `$X` in the entry causes any comments in the list file (such as notes about who subscribed an address and when) to be stripped, by default, when sending the list file.

⓫ This entry allows `open-list-owner@example.com` to use the command `PURGE/LIST` open-list to cause comment lines (such as addresses unsubscribed via `MAILSERV`) to be removed from the list file.

⓬ This entry allows `open-list-owner@example.com` to subscribe other addresses to the `open-list-reject` list. The `$K` means that the subscriptions will not be immediately performed, but rather `MAILSERV` will send back a message to (due to the `$V`) `open-list-owner@example.com` asking him to confirm the subscription. The `$D` means that open-list-owner can use the `/NONOTIFY` qualifier on subscribe requests, *e.g.*, SUBSCRIBE/NONOTIFY open-list-reject, to cause omission of the usual "You have been subscribed by open-list-owner@example.com to open-list-reject list" message to the subscribee address.

⓭ This entry disallows all subscribe access (other than that explicitly allowed earlier in ⓬) to the open-list-reject list.

⓮ This entry allows open-list-owner@example.com to unsubscribe other addresses from the `open-list-reject` list. The `$K` means that the unsubscriptions will not be immediately performed, but rather `MAILSERV` will send back a message to (due to the `$V`) `open-list-owner@example.com` asking him to confirm the unsubscription. The `$D` means that open-list-owner can use the `/NONOTIFY` qualifier on unsubscribe requests, *e.g.*, `UNSUBSCRIBE/NONOTIFY` open-list-reject, to cause omission of the usual "You have been unsubscribed by open-list-owner@example.com from open-list-reject list" message to the unsubscribee address.

⓯ This entry disallows all unsubscribe access (other than that explicitly allowed earlier in ⓮) to the `open-list-reject` list.

⓰ This entry allows `open-list-owner@example.com` to use the command `SEND/LIST open-list-reject` to request a copy of the `open-list-reject` list.

⓱ This entry allows `open-list-owner@example.com` to use the command `PURGE/LIST open-list-reject` to purge commented lines (including addresses unsubscribed via `MAILSERV`) from the `open-list-reject` list.

⓲ This entry causes `open-list-reject` to *not* be displayed as a list in response to any `DIRECTORY/LIST` command.

⓳ This entry disallows any other `MAILSERV` access to the `open-list-reject` list.

⓴ This `MAILSERV_LISTS` entry specifies that `MAILSERV` responses to user messages to MAILSERV regarding `open-list` will have a `From:` address of `open-list-owner@example.com`.

4.4.2 A Semi-restricted Mailing List

This example will discuss an example of a somewhat restricted list. Subscription requests are referred to a list owner. Only members of the list can post directly to the list; if others attempt to post, their postings will be referred to the list owner. Such a list definition in the PMDF alias file might be:

```
group-list:         group-list-expand@process
group-list-expand: <PMDF_MAILSERV_MAIL_DIR:group-list.dis, \
                    [MODERATOR_ADDRESS] group-list-owner@example.com, \
                    [MODERATOR_LIST] PMDF_MAILSERV_MAIL_DIR:group-list.dis, \
                    [USERNAME] group-list-owner, \
                    [HEADER_ADDITION] PMDF_TABLE:group-list-headers.txt, \
                    group-list-owner@example.com, \
                    group-list@example.com
group-list-request: MAILSERV
```

Because the entire list is used for the `[MODERATOR_LIST]` argument, anyone already on the list can post directly to the list. But attempted postings from anyone else will be referred to the `[MODERATOR_ADDRESS]` argument, `group-list-owner@example.com`

The above list definition references a file of headers to be added to messages posted to the list; such a file might be:

```
List-Help: <mailto:mailserv@example.com?body=help> (MAILSERV Instructions),
     <mailto:group-list-owner@example.com?subject=help> (List Manager)
List-Subscribe:
     <mailto:group-list-owner@example.com?subject=subscribe%20group-list>
List-Unsubscribe:
     <mailto:mailserv@example.com?body=unsubscribe%20group-list>
List-Post: <mailto:group-list-owner@example.com> (List Moderator)
List-Owner: <mailto:group-list-owner@example.com?Subject=group-list>
```

Only `group-list-owner@example.com` will be permitted to subscribe users to the list. `group-list-owner` can also unsubscribe other users from the list. And `group-list-owner@example.com` will have to confirm any `SUBSCRIBE` or `UNSUBSCRIBE` commands; that is, when `group-list-owner@example.com` sends a `SUBSCRIBE` or `UNSUBSCRIBE` request to `MAILSERV`, `MAILSERV` will send back to `group-list-owner@example.com` a message containing a cookie string, which `group-list-owner@example.com` will need to include in a second, confirming message in order for the command to actually be performed. Users can unsubscribe themselves, but attempts to unsubscribe others will be referred to the `group-list-owner`. Only members of the list will be permitted to request the list membership file, and `MAILSERV` will require that such requests be confirmed with a second request message including a cookie string that `MAILSERV` sends out to the supposed requestor address.

```
MAILSERV_ACCESS
```

```
  SUBSCRIBE|group-list|group-list-owner@example.com|*              $K$V$Y  ❶
  SUBSCRIBE|group-list|group-list-owner@example.com                  $K$Y  ❷
  SUBSCRIBE|group-list|*|*                 $*group-list-owner@example.com  ❸
  SUBSCRIBE|group-list|*                   $*group-list-owner@example.com  ❹
  UNSUBSCRIBE|group-list|group-list-owner@example.com|*            $K$V$Y  ❺
  UNSUBSCRIBE|group-list|group-list-owner@example.com                $K$Y  ❻
  UNSUBSCRIBE|group-list|*|*            $Y$*group-list-owner@example.com  ❼
  UNSUBSCRIBE|group-list|*                                             $Y  ❽
  SENDLIST|group-list|group-list-owner@example.com                     $Y  ❾
  SENDLIST|group-list|*     $X$<PMDF_MAILSERV_MAIL_DIR:group-list.dis      ❿
  PURGELIST|group-list|group-list-owner@example
MAILSERV_LISTS

  group-list                               $Agroup-list-owner@example.com  ⓫
```

More specifically:

❶ This entry explicitly allows `group-list-owner@example.com` to subscribe others to the list; a later entry, ❸, will cause third-party subscribe attempts from any other sending address to be redirected to `group-list-owner@example.com`. Note the use of $K; this means that any subscription will not be performed immediately, but rather `MAILSERV` will send back a challenge message to (due to the `$V` `group-list-owner@example.com` asking him to confirm the subscription.

❷ This entry explicitly allows `group-list-owner@example.com` to subscribe himself to the list; a later entry, ❹, will cause direct subscribe attempts from any other sending address to be redirected to `group-list-owner@example.com`. Note the use of $K; this means that such a subscription will not be performed immediately, but rather `MAILSERV` will send back a challenge message to `group-list-owner@example.com` asking him to confirm the subscription.

❸ This entry causes third party subscribe attempts to be redirected to `group-list-owner@example.com`.

❹ This entry causes direct subscribe attempts to be redirected to `group-list-owner@example.com`.

❺ This entry allows `group-list-owner@example.com` to unsubscribe other addresses from the list. Note the use of `$K`; this means that any unsubscription will not be performed immediately, but rather `MAILSERV` will send back a challenge message to (due to the `$V`) `group-list-owner@example.com` asking him to confirm the unsubscription.

❻ This entry allows `group-list-owner@example.com` to unsubscribe himself from the list. Note the use of `$K`; this means that any unsubscription will not be performed immediately, but rather `MAILSERV` will send back a challenge message to `group-list-owner@example.com` asking him to confirm the unsubscription.

❼ This entry causes third-party unsubscribe attempts to be redirected to `group-list-owner@example.com`. Since third party unsubscribes are disallowed by default, an alternative would be to not put in any entry and get the default behavior of disallowing them. But it can be useful for `group-list-owner` to know about attempted third party unsubscribes; for instance, users whose addresses change can be attempting to unsubscribe their old addresses.

❽ This entry allows general users to unsubscribe themselves. This is allowed by default, so strictly speaking this entry is not necessary; however, it is included for completeness and clarity.

❾ This entry allows `group-list-owner@example.com` to get a copy of the list membership file.

❿ This entry allows members of group-list to get a copy of the list membership file; because of the `$X` in the entry, they will not get comment lines by default.

⓫ This entry allows `group-list-owner@example.com` to purge the list membership file of comment lines (such as `MAILSERV` comment lines showing unsubscribed users).

⓬ This `MAILSERV_LISTS` entry specifies that `MAILSERV` responses to user messages to `MAILSERV` regarding group-list will have a `From:` address of `group-list-owner@example.com`.

5 The Mapping File

Many components of PMDF employ table lookup oriented information. One particular type of table is used more often in PMDF than any other. Generally speaking, this sort of table is used to transform (*i.e.,* map) an input string into an output string. Such tables, referred to as mapping tables, are usually presented as two columns, the first or left-hand column giving the possible input strings and the second or right-hand column giving the resulting output string for the input it is associated with. Most of the PMDF databases are instances of just this sort of mapping table. PMDF database files, however, do not provide wildcard lookup facilities, owing to inherent inefficiencies in having to scan the entire database for wildcard matches.

The mapping file provides PMDF with facilities for supporting multiple mapping tables. Full wildcard facilities are provided, and multi-step and iterative mapping methods can be accommodated as well. This approach is more compute-intensive than using a database, especially when the number of entries is large. However, the attendant gain in flexibility can actually serve to eliminate the need for most of the entries in an equivalent database, and this can actually result in lower overhead overall.

A complete list of the mapping table names recognized by PMDF is provided under the index entry "Mappings". You can test mapping tables with the `PMDF TEST/MAPPING` (OpenVMS) or `pmdf test -mapping` (UNIX and NT) utility. See Chapter 29 and Chapter 30 for details.

5.1 Locating and Loading the Mapping File

All mappings are kept in the PMDF mapping file.[1] Each time a PMDF program begins running, this file is read and loaded into memory. This overhead can be avoided by compiling your PMDF configuration, in which case the contents of the mapping file will be incorporated into the compiled configuration. The disadvantage to this, however, is that it means that the configuration must be recompiled and reinstalled whenever a change is made to the mapping file or to an include file used by the mapping file. See Section 8.1 for details on compiling your configuration.

The mapping file should be world readable. Failure to allow world read access will lead to erratic behavior.

[1] This is the file pointed to by the `PMDF_MAPPING_FILE` logical (OpenVMS), PMDF Tailor file entry (UNIX), or Registry entry (NT), and hence is usually `PMDF_TABLE:mappings.` (OpenVMS) or `/pmdf/table/mappings` (UNIX) or `C:\pmdf\table\mappings` (NT).

5.2 File Format

The mapping file consists of a series of separate tables. Each table begins with its name. Names always have an alphabetic character in the first column. The table name is followed by a required blank line, and then by the entries in the table. Entries consist of zero or more indented lines. Each entry line consists of two columns separated by one or more spaces or tabs. Any spaces within an entry must be quoted; see Section 5.3.1. It is required that a blank line appear after each mapping table name and between each mapping table; no blank lines can appear between entries in a single table. Comments are introduced by an exclamation mark, `!`, appearing in the first column.

Pictorially, the format that results looks like this:

```
TABLE-1-NAME

   pattern1-1    template1-1
   pattern1-2    template1-2
   pattern1-3    template1-3
     .             .
     .             .
     .             .
   pattern1-n    template1-n

TABLE-2-NAME

   pattern2-1    template2-1
   pattern2-2    template2-2
   pattern2-3    template2-3
     .             .
     .             .
     .             .
   pattern2-n    template2-n

           .
           .
           .

TABLE-m-NAME

           .
           .
           .
```

In this example an application using the mapping table *TABLE-2-NAME* would map the string *pattern2-2* into whatever is specified by *template2-2*. Each pattern or template can contain up to 252 characters. There is no limit to the number of entries that can appear in a mapping (although excessive numbers of entries can eat up huge amounts of CPU and can consume excessive amounts of memory). Long lines can be continued by ending them with a backslash, (\). The white-space between the two columns and before the first column can not be omitted.

Duplicate mapping table names are not allowed in the mapping file.

5.2.1 Including Other Files in the Mapping File

Other files can be included in the mapping file. This is done with a line of the form:

```
<file-spec
```

This will effectively substitute the contents of the file *file-spec* into the mapping file at the point where the include appears. The file specification should specify a full file path (device, directory, *etc.*). All files included in this fashion must be world readable. Comments are also allowed in such included mapping files. Includes can be nested up to three levels deep. Include files are loaded at the same time the mapping file is loaded — they are not loaded on demand, so there is no performance or memory savings involved in using include files.

5.3 Mapping Operations

All mappings in the mapping file are applied in a consistent way. The only things that change from one mapping to the next is the source of input strings and what the output from the mapping is used for.

A mapping operation always starts off with an input string and a mapping table. The entries in the mapping table are scanned one at a time from top to bottom in the order in which they appear in the table. The left hand side of each entry is used as pattern and the input string is compared in a case-blind fashion with that pattern.

5.3.1 Mapping Entry Patterns

Patterns can contain wildcard characters. In particular, the usual OpenVMS wildcard characters are allowed: an asterisk, `*`, will match zero or more characters and each percent sign, `%`, will match a single character. Asterisks, percent signs, spaces, and tabs can be quoted by preceeding them with a dollar sign, `$`. Quoting an asterisk or percent sign robs it of any special meaning. Spaces and tabs must be quoted to prevent them from ending prematurely a pattern or template. Literal dollar sign characters should be doubled, `$$`, the first dollar sign quoting the second one. Additional wildcards available in patterns are listed in Table 5–1.

Table 5–1 Mapping Pattern Wildcards

Wildcard	Description
%	Match exactly one character
*	Match zero or more characters, with maximal or ‘‘greedy’’ left-to-right matching
Back match	**Description**
$*n**	Match the *n*th wildcard or glob

Table 5–1 (Cont.) Mapping Pattern Wildcards

Modifiers	Description
$_	Use minimal or "lazy" left-to-right matching
$@	Turn off "saving" of the succeeding wildcard or glob
$^	Turn on "saving" of the succeeding wildcard or glob; this is the default

Glob wildcard	Description
$A%	Match one alphabetic character, A–Z or a–z
$A*	Match zero or more alphabetic characters, A–Z or a–z
$B%	Match one binary digit (0 or 1)
$B*	Match zero or more binary digits (0 or 1)
$D%	Match one decimal digit 0–9
$D*	Match zero or more decimal digits 0–9
$H%	Match one hexadecimal digit 0–9 or A–F
$H*	Match zero or more hexadecimal digits 0–9 or A–F
$O%	Match one octal digit 0–7
$O*	Match zero or more octal digits 0–7
$S%	Match one symbol set character, *i.e.*, 0–9, A–Z, a–z, _, $
$S*	Match zero or more symbol set characters, *i.e.*, 0–9, A–Z, a–z, _, $
$T%	Match one tab or vertical tab or space character
$T*	Match zero or more tab or vertical tab or space characters
$X%	A synonym for $H%
$X*	A synonym for $H*
$[*c*]%	Match character *c*
$[*c*]*	Match arbitrary occurrences of character *c*
$[$c_1 c_2 ... c_n$]%	Match exactly one occurrence of character c_1, c_2, ..., or c_n
$[$c_1 c_2 ... c_n$]*	Match arbitrary occurrences of any characters c_1, c_2, ..., or c_n
$[$c_1$-$c_n$]%	Match any one character in the range c_1 to c_n
$[$c_1$-$c_n$]*	Match arbitrary occurrences of characters in the range c_1 to c_n
$<*IPv4*>	Match an IPv4 address, ignoring bits
$(*IPv4*)	Match an IPv4 address, keeping prefix bits
${*IPv6*}	Match an IPv6 address

Note that to specify multiple modifiers, or to specify modifiers and a back match, the syntax uses just one dollar character. For instance, to back match the initial wild card, without saving the back match itself, one would use `$@0`, *not* `$@$0`.

Note that the `PMDF TEST/MATCH` (OpenVMS) or `pmdf test -match` (UNIX) utility can be used to test mapping patterns and specifically to test wildcard behavior in patterns.

5.3.1.1 The $_ modifier: minimal *vs.* maximal Matching

Asterisk, `*`, wildcards maximize what they match working from left to right across the pattern. For instance, when the string `a/b/c` is compared to the pattern `*/*`, the left asterisk will match `a/b` and the right asterisk will match `c`.

The `$_` modifier causes wildcard matching to be minimized, where the least possible match is considered the match, working from left to right across the pattern. For instance, when the string `a/b/c` is compared to the pattern `$_*/$_*`, the left `$_*` will match `a` and the right `$_*` will match `b/c`.

5.3.1.2 IP Matching

With IPv4 "prefix bits" matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits from the prefix that are significant when comparing for a match. For instance,

```
$(123.45.67.0/24)
```

will match anything in the 123.45.67.0 subnet.

With IPv4 "ignore bits" matching, an IP address or subnet is specified, optionally followed by a slash and the number of bits to ignore when checking for a match. For instance,

```
$<123.45.67.0/8>
```

will match anything in the 123.45.67.0 subnet. Or another example is that

```
$<123.45.67.4/2>
```

will match anything in the range 123.45.67.4–123.45.67.7.

IPv6 matching matches an IPv6 address or subnet.

5.3.1.3 Character Matching

Within globs, *i.e.*, within a `$[...]` construct, the backslash character, (`\`), is the quote character. To represent a literal hyphen, `-`, or right bracket, `]`, within a glob the hyphen or right bracket must be quoted with a backslash.

All other characters in a pattern just represent and match themselves. In particular, single and double quote characters as well as parentheses have no special meaning in either mapping patterns or templates; they are just ordinary characters. This makes it easy to write entries that correspond to illegal addresses or partial addresses.

Also note that within a single `$[...]` construct, there can be multiple ranges of characters ($c_1c_2...c_n$) and multiple lists of characters (c_1-c_n) mixed and matched.

5.3.2 Mapping Entry Templates

If the comparison of the pattern in a given entry fails, no action is taken; the scan proceeds to the next entry. If the comparison succeeds, the right hand side of the entry is used as a template to produce an output string. The template effectively causes the replacement of the input string with the output string that is constructed from the instructions given by the template.

Almost all characters in the template simply produce themselves in the output. The one exception is a dollar sign.

A dollar sign followed by a dollar sign, space, or tab produces a dollar sign, space, or tab in the output string. Note that all these characters must be quoted in order to be inserted into the output string.

A dollar sign followed by a digit *N* calls for a substitution; a dollar sign followed by an alphabetic character is referred to as a "metacharacter". Metacharacters themselves will not appear in the output string produced by a template. See Table 5–2 for a list of the special substitution and standard processing metacharacters. Any other metacharacters are reserved for mapping-specific applications.

Note that any of the metacharacters `$C`, `$E`, `$L`, or `$R`, when present in the template of a matching pattern, will influence the mapping process, controlling whether it terminates or continues. That is, it is possible to set up iterative mapping table entries, where the output of one entry becomes the input of another entry. If the template of a matching pattern does not contain any of the metacharacters `$C`, `$E`, `$L`, or `$R`, then `$E` (immediate termination of the mapping process) is assumed.

The number of iterative passes through a mapping table is limited to prevent infinite loops. A counter is incremented each time a pass is restarted with a pattern that is the same length or longer than the previous pass. If the string has a shorter length than previously, the counter is reset to zero. A request to again iterate a mapping is not honored after the counter has exceeded 10.

Table 5–2 Mapping Template Substitutions and Metacharacters

Substitution sequence	Section	Substitutes
$*n*	5.3.2.1	*n*th wildcarded field as counted from left to right starting from 0
$#...#	5.3.2.6	Sequence number substitution.
$]...[	5.3.2.7	LDAP search URL lookup; substitute in result.
${...}	5.3.2.8	General database substitution.
$|...|	5.3.2.9	Apply specified mapping table to supplied string.
$[...]	5.3.2.10	Invoke site-supplied routine; substitute in result.

Table 5–2 (Cont.) Mapping Template Substitutions and Metacharacters

Metacharacter	Section	Description
$C	5.3.2.3	Continue the mapping process starting with the next table entry; use the output string of this entry as the new input string for the mapping process.
$E	5.3.2.3	End the mapping process now; use the output string from this entry as the final result of the mapping process.
$L	5.3.2.3	Continue the mapping process starting with the next table entry; use the output string of this entry as the new input string; after all entries in the table are exhausted make one additional pass starting with the first table entry. A subsequent match can override this condition with a $C, $E, or $R metacharacter.
$R	5.3.2.3	Continue the mapping process starting with the first entry of the the mapping table; use the output string of this entry as the new input string for the mapping process.
$?*x*?	5.3.2.5	Mapping entry succeeds *x* percent of the time.
$\	5.3.2.2	Force subsequent text to lowercase.
$^	5.3.2.2	Force subsequent text to uppercase.
$_	5.3.2.2	Leave subsequent text in its original case.
$:*x*	5.3.2.4	Match only if the specified flag is set.
$;*x*	5.3.2.4	Match only if the specified flag is clear.

5.3.2.1 Wildcard Field Substitutions, $*n*

A dollar sign followed by a digit *n* is replaced with the material that matched the *n*th wildcard in the pattern. The wildcards are numbered starting with 0. For example, the entry

```
PSI$%*::*     $1@$0.psi.example.com
```

would match the input string `PSI%A::B` and produce the resultant output string `b@a.psi.example.com`. The input string `PSI%1234::USER` would also match producing `USER@1234.psi.example.com` as the output string. The input string `PSIABC::DEF` would not match the pattern in this entry and no action would be taken; *i.e.,* no output string would result from this entry.

5.3.2.2 Controlling Text Case, $\, $^, $_

`$\` forces subsequent text to lowercase, `$^` forces subsequent text to uppercase, and `$_` causes subsequent text to retain its original case. For instance, these metacharacters can be useful when using mappings to transform addresses for which case is significant.

5.3.2.3 Processing Control, $C, $L, $R, $E

The $C, $L, $R, and $E metacharacters influence the mapping process, controlling whether and when the mapping process terminates. $C causes the mapping process to continue with the next entry, using the output string of the current entry as the new input string for the mapping process. $L causes the mapping process to continue with the next entry, using the output string of the current entry as the new input string for the mapping process, and, if no matching entry is found, making one additional pass through the table starting with the first table entry; a subsequent matching entry with a $C, $E, or $R metacharacter overrides this condition. $R causes the mapping process to continue from the first entry of the table, using the output string of the current entry as the new input string for the mapping process. $E causes the mapping process to terminate; the output string of this entry is the final output. $E is the default.

Mapping table templates are scanned left to right. So to set a $C, $L, or $R flag for entries that can "succeed" or "fail", *e.g.*, general database substitutions, or random value controlled entries, put the $C, $L, or $R metacharacter to the left of the part of the entry that can succeed or fail; otherwise, if the remainder of the entry fails, the flag will not be seen.

5.3.2.4 Check for Special Flags

Some mapping probes have special flags set. $:x causes an entry to match only if the flag x is set. $;x causes an entry to match only if the flag x is clear. See specific mapping table descriptions for any special flags that can apply for that table.

When the intention is that an entry should succeed and terminate if the flag check succeeds, but that the mapping process should continue if the flag check fails, then the entry should use the $C metacharacter to the left of the flag check and use the $E flag to the right of the flag check.

5.3.2.5 Entry Randomly Succeeds or Fails, $?*x*?

$?x? in a mapping table entry causes the entry to "succeed" x percent of the time; the rest of the time, the entry "fails" and the output of the mapping entry's input is taken unchanged as the output. (Note that, depending upon the mapping, the effect of the entry "failing" is not necessarily the same as the entry not matching in the first place.) The argument between the ?'s, x, should consist of a real number specifying the success percentage.

For instance, suppose that a system with IP address 123.45.6.78 is sending your site just a little too much e-mail and you'd like to slow it down; if you're using the multithreaded TCP SMTP channel, you can use a PORT_ACCESS mapping table in the following way. Suppose you'd like to allow through only 25 percent of its connection attempts and temporarily reject the other 75 percent of its connection attempts. The following PORT_ACCESS mapping table uses $?25? to cause the entry with the $Y (accept the connection) to succeed only 25 percent of the time; the other 75 percent of the time, when this entry fails, the initial $C on that entry causes PMDF to continue the mapping from the next entry, which causes the connection attempt to be rejected with a temporary SMTP error (in this example, 452 4.4.0) and the text message "Try again later".

```
PORT_ACCESS

  TCP|*|25|123.45.6.78|*      $C$?25?$Y
  TCP|*|25|123.45.6.78|*      $N452$ 4.4.0$ Try$ again$ later
```

Or to similarly reject many, but not all, POP3 and IMAP connection attempts from a system with IP address 123.45.6.78:

```
PORT_ACCESS

! POP3 connections
  TCP|*|110|123.45.6.78|*     $C$?25?$Y
  TCP|*|110|123.45.6.78|*     $N-ERR$ Try$ again$ later
! IMAP connections
  TCP|*|143|123.45.6.78|*     $C$?25?$Y
  TCP|*|143|123.45.6.78|*     $N*$ BYE$ Try$ again$ later
```

See Section 21.2.1 for details on the PORT_ACCESS mapping table.

Another example would be to randomly issue a temporary failure message for a certain percentage of SMTP messages from a particular envelope From: address; for instance, suppose the goal is to issue a temporary failure message with extended SMTP code 4.5.9 to 80 percent of the messages that busybee@some.where attempts to send to your local channel users. Then a SEND_ACCESS mapping table could be used, *e.g.*,

```
SEND_ACCESS

  tcp_*|busybee@some.where|1|*     $C$?20?$Y
  tcp_*|busybee@some.where|1|*     $N$X4.5.9|Try$ again$ later
```

5.3.2.6 Sequence Number Substitutions, $#...#

A $#...# substitution increments the value stored in a PMDF sequence file and substitutes that value into the template. This can be used to generate unique, increasing strings in cases where it is desirable to have a uniquifier in the mapping table output; for instance, when using a mapping table to generate file names.

Permitted syntax is any one of:

$#*seq-file-spec*|*radix*|*width*#

or

$#*seq-file-spec*|*radix*#

or

$#*seq-file-spec*#

where the optional *seq-file-spec* argument is a full file specification for an (already existing) PMDF sequence file, and where the optional *radix* and *width* arguments specify the radix (base) in which to output the sequence value, and the number of digits to output, respectively. The *seq-file-spec* argument can be omitted, in which case PMDF will use its own temporary sequence file (that will be created and used for the duration of this image). The default radix is 10. Radices in the range -36 to 36 are also allowed; for instance, base 36 gives values expressed with digits 0,...,9,A,...,Z. By default,

the sequence value is printed in its natural width, but if the specified width calls for a greater number of digits, then the output will be padded with 0's on the left to obtain the desired number of digits. Note that if a width is explicitly specified, then the radix must be explicitly specified also.

As noted above, when specifying an explicit PMDF sequence file in a mapping, that file must already exist. To create a PMDF sequence file, on OpenVMS use the command

```
$ CREATE/FDL=PMDF_COM:sequence_number.fdl seq-file-spec
```

or on UNIX use the command

```
% touch seq-file-spec
```

or

```
% cat >seq-file-spec
```

or on NT use the command

```
C:\> copy nul seq-file-spec
```

A sequence number file accessed via a mapping table must be world readable in order to operate properly. You must also have a PMDF user account in order to use such sequence number files.

VMS

On OpenVMS, if you did not create a PMDF user account PMDF_USER during your PMDF installation, you should use the PMDF_COM:create_pmdf_user_account.com procedure to create it.

5.3.2.7 LDAP Query URL Substitutions, $]...[

A substitution of the form $]*ldap-url*[is handled specially. *ldap-url* is interpreted as an LDAP query URL and the result of the LDAP query is substituted. Standard LDAP URLs are used, with the host and port omitted; the host and port are instead specified with the LDAP_HOST and LDAP_PORT PMDF options (see Section 7.3.2 for further discussion of this option). That is, the LDAP URL should be specified as

```
ldap:///dn[?attributes[?scope?filter]]
```

where the square bracket characters *[* and *]* shown above indicate optional portions of the URL. The *dn* is required and is a distinguished name specifying the search base. The optional *attributes*, *scope*, and *filter* portions of the URL further refine what information to return. That is, *attributes* specifies the attribute or attributes to be returned from LDAP directory entries matching this LDAP query. The *scope* can be any of base (the default), one, or sub. *filter* describes the characteristics of matching entries.

Certain LDAP URL substitution sequences are available for use within the LDAP query URL; see Table 3–1 for a full list.

5.3.2.8 General Database Substitutions, ${...}

A substitution of the form ${*text*} is handled specially. The *text* part is used as a key to access the PMDF general database.[2] This database is generated with the `PMDF CRDB` (OpenVMS) or `pmdf crdb` (UNIX and NT) utility. If *text* is found in the database, the corresponding template from the database is substituted. If *text* does not match an entry in the database, the input string is used unchanged as the output string.

If a general database exists, it should be world readable to ensure that it operates properly.

Note that when wanting to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a general database substitution, the processing control metacharacter should be placed to the left of the general database substitution in the mapping table template; otherwise the "failure" of a general database substitution will mean that the processing control metacharacter will not be seen.

5.3.2.9 Mapping Table Substitutions, $|...|

A substitution of the form $ | *mapping*;*argument* | is handled specially. PMDF looks for an auxiliary mapping table named *mapping* in the PMDF mapping file, and uses *argument* as the input to that named auxiliary mapping table. The named auxiliary mapping table must exist and must set the `$Y` flag in its output if it is successful; if the named auxiliary mapping table does not exist or doesn't set the `$Y` flag, then that auxiliary mapping table substitution fails and the original mapping entry is considered to fail: the original input string will be used as the output string.

Note that when wanting to use processing control metacharacters such as `$C`, `$R`, or `$L` in a mapping table entry that does a mapping table substitution, the processing control metacharacter should be placed to the left of the mapping table substitution in the mapping table template; otherwise the "failure" of a mapping table substitution will mean that the processing control metacharacter will not be seen.

5.3.2.10 Site-supplied Routine Substitutions, $[...]

A substitution of the form `$[`*`image,routine,argument`*`]` is handled specially. The *`image,routine,argument`* part is used to find and call a customer-supplied routine. At run-time on OpenVMS, PMDF uses `LIB$FIND_IMAGE_SYMBOL` to dynamically load and link to the routine *`routine`* from the shareable image *`image`*; at run-time on UNIX, PMDF uses `dlopen` and `dlsym` to dynamically load and call the routine *`routine`* from the shared library *`image`*; at run-time on NT, PMDF calls the routine *`routine`* from the dynamic link library *`image`*. The routine *`routine`* is then called as a function with the following argument list:

[2] The PMDF general database is referenced via the `PMDF_GENERAL_DATABASE` logical (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT). Hence, by default, the general database is `PMDF_TABLE:general.dat` (OpenVMS) or the file `/pmdf/table/generaldb.*` (UNIX) or the file `C:\pmdf\table\generaldb.*` (NT).

status := *routine* (**argument**, **arglength**, **result**, **reslength**)

argument and **result** are 252 byte long character string buffers. On OpenVMS, **argument** and **result** are passed by descriptor (a class S descriptor is used to ensure maximum compatibility); on UNIX and NT, **argument** and **result** are passed as a pointer to a character string, (*e.g.*, in C, as char*.) **arglength** and **reslength** are signed, long integers passed by reference. On input, **argument** contains the *argument* string from the mapping table template, and **arglength** the length of that string. On return, the resultant string should be placed in **result** and its length in **reslength**. This resultant string will then replace the "$[*image,routine,argument*]" in the mapping table template. The routine *routine* should return 0 if the mapping table substitution should fail and -1 if the mapping table substitution should succeed. If the substitution fails, then normally the original input string will be used unchanged as the output string.

Note that when wanting to use processing control metacharacters such as $C, $R, or $L in a mapping table entry that does a site-supplied routine substitution, the processing control metacharacter should be placed to the left of the site-supplied routine substitution in the mapping table template; otherwise the "failure" of a mapping table substitution will mean that the processing control metacharacter will not be seen.

The site-supplied routine callout mechanism allows PMDF's mapping process to be extended in all sorts of complex ways. For example, in a PORT_ACCESS or SEND_ACCESS mapping table, a call to some type of load monitoring service could be performed and the result used to decide whether or not to accept a connection or message.

VMS

On OpenVMS systems, since LIB$FIND_IMAGE_SYMBOL is used to dynamically load the site-supplied image *image*, then *image* must be a logical name pointing to the actual shareable image. Moreover, as this mechanism will be invoked by PMDF in a variety of contexts, the logical must be an executive mode logical, any logicals it references must also be executive mode logicals, and the image itself must be world readable and installed as a known image.

UNIX

On UNIX systems, the site-supplied shared library image *image* should be world readable.

Note: This facility is not designed for use by casual users; it is intended to be used to extend PMDF's capabilities system-wide.

5.3.3 A Complex Mapping Example

Example 5–1 shows a mapping that will take a roman numeral as input and will output the equivalent decimal integer. Although this example is completely contrived, it does show almost all of the features of the mapping facility. Multiple passes over the patterns are used, as well as continuations and substitutions.

Example 5–1 Mapping File Example

```
ROMAN-TO-INTEGER

  *|0|%*                Input$ Error$E
  0%*|0|                $0$1|0|$R
  *|0|                  Value$ =$ $0$E
  *|4|MMM*              $03|3|$1$C
  *|4|MM*               $02|3|$1$C
  *|4|M*                $01|3|$1$C
  *|4|*                 $00|3|$1$C
  *|3|CM*               $09|2|$1$C
  *|3|DCCC*             $08|2|$1$C
  *|3|DCC*              $07|2|$1$C
  *|3|DC*               $06|2|$1$C
  *|3|D*                $05|2|$1$C
  *|3|CD*               $04|2|$1$C
  *|3|CCC*              $03|2|$1$C
  *|3|CC*               $02|2|$1$C
  *|3|C*                $01|2|$1$C
  *|3|*                 $00|2|$1$C
  *|2|XC*               $09|1|$1$C
  *|2|LXXX*             $08|1|$1$C
  *|2|LXX*              $07|1|$1$C
  *|2|LX*               $06|1|$1$C
  *|2|L*                $05|1|$1$C
  *|2|XL*               $04|1|$1$C
  *|2|XXX*              $03|1|$1$C
  *|2|XX*               $02|1|$1$C
  *|2|X*                $01|1|$1$C
  *|2|*                 $00|1|$1$C
  *|1|IX*               $09|0|$1$R
  *|1|VIII*             $08|0|$1$R
  *|1|VII*              $07|0|$1$R
  *|1|VI*               $06|0|$1$R
  *|1|V*                $05|0|$1$R
  *|1|IV*               $04|0|$1$R
  *|1|III*              $03|0|$1$R
  *|1|II*               $02|0|$1$R
  *|1|I*                $01|0|$1$R
  *|1|*                 $00|0|$1$R
  *                     |4|$0$R
```

The operation of this example mapping is best understood by tracing an input string through the process. For instance, suppose the input string `CDLXIV` is given. On the first pass through, the very last pattern, `*`, is the one that matches — it produces the output string `|4|CDLXIV` and, because of the `$R` specified in the template, resumes the mapping process from the start of the table using the output string `|4|CDLXIV` as the new input string.

On the second pass through, the pattern that matches first is `*|4|*`, which produces the new string `0|3|CDLXIV`. Scanning continues from this point because of the `$C` sequence. The next pattern that matches is `*|3|CD*`, which produces the string `04|2|LXIV`. Processing continues and the pattern `*|2|LX*` matches, producing `046|1|IV`. Processing continues and the pattern `*|1|IV*` matches, producing `0464|0|`, and the `$R` takes the process back to the top of the table.

On the third pass through, the pattern that matches first is `0%*|0|`, which produces `464|0|` and restarts at the top again.

On the fourth pass through, the pattern that matches first is `*|0|`, which produces the string "`Value = 464`" and exits because of the `$E`.

6 Character Set Conversions and Message Reformatting

6.1 CHARSET-CONVERSION Mapping Table

One very basic mapping table in PMDF is the character set conversion table. The name of this table is CHARSET-CONVERSION. It is used to specify what sorts of channel-to-channel character set conversions, labelling conversions, and message reformatting should be done.

On many systems there is no need to do character set conversions or message reformating and therefore this table is not needed. Situations arise, however, where character conversions must be done. For example, sites running Japanese OpenVMS may need to convert between DEC Kanji and the ISO-2022 Kanji currently used on the Internet. Another possible use of conversions arises when multinational characters are so heavily used that the slight discrepancies between the DEC Multinational Character Set (DEC-MCS) and the ISO-8859-1 character set specified for use in MIME may become an issue, and actual conversion between the two may therefore be needed.

The CHARSET-CONVERSION mapping can also be used to alter the format of messages. Facilities are provided to convert a number of non-MIME formats into MIME. Changes to MIME encodings and structure are also possible. These options are used when messages are being relayed to systems that only support MIME or some subset of MIME. And finally, conversion from MIME into non-MIME formats is provided in a small number of cases.

PMDF will probe the CHARSET-CONVERSION mapping table in two different ways. The first probe is used to determine whether or not PMDF should reformat the message and if so, what formatting options should be used. (If no reformatting is specified PMDF does not bother to check for specific character set conversions.) The input string for this first probe has the general form:

IN-CHAN=*in-channel*;OUT-CHAN=*out-channel*;CONVERT

Here *in-channel* is the name of the source channel (where the message comes from) and *out-channel* is the name of the destination channel (where the message is going). If a match occurs the resulting string should be a comma-separated list of keywords. Table 6–1 lists the available keywords.

Note: Make sure that there is no whitespace in the resulting string, for example around commas or equal signs.

Table 6–1 CHARSET-CONVERSIONS Mapping Table Keywords

Keyword	Action
Always	Force conversion even when conversion channel is an intermediate destination
Appledouble	Convert other MacMIME formats to Appledouble format

Table 6–1 (Cont.) CHARSET-CONVERSIONS Mapping Table Keywords

Keyword	Action
Applesingle	Convert other MacMIME formats to Applesingle format
BASE64	Switch MIME encodings to BASE64
Binhex	Convert other MacMIME formats, or parts including Macintosh type and Mac creator information, to Binhex format
Block	Extract just the data fork from MacMIME format parts
Bottom	"Flatten" any message/rfc822 body part (forwarded message) into a message content part and a header part
Delete	"Flatten" any message/rfc822 body part (forwarded message) into a message content part, deleting the forwarded headers
Level	Remove redundant multipart levels from message
Macbinary	Convert other MacMIME formats, or parts including Macintosh type and Macintosh creator information, to Macbinary format
No	Disable conversion
Pathworks	Convert message to Pathworks Mail format
QUOTED-PRINTABLE	Switch MIME encodings to QUOTED-PRINTABLE
Record,Text	Line wrap text/plain parts at 80 characters
Record,Text=*n*	Line wrap text/plain parts at *n* characters
RFC1154	Convert message to RFC 1154 format
Top	"Flatten" any message/rfc822 body part (forwarded message) into a header part and a message content part
UUENCODE	Switch MIME encodings to X-UUENCODE
Yes	Enable conversion

A `No` is assumed if no match occurs.

6.2 Character Set Conversion

If PMDF probes and finds that the message is to be reformatted, it will proceed to check each part of the message. Any text parts are found and their character set parameters are used to generate the second probe. Only when PMDF has checked and found that conversions may be needed does it ever perform the second probe. The input string in this second case looks like this:

`IN-CHAN=`*`in-channel`*`;OUT-CHAN=`*`out-channel`*`;IN-CHARSET=`*`in-char-set`*

The *`in-channel`* and *`out-channel`* are the same as before, and the *`in-char-set`* is the name of the character set associated with the particular part in question. If no match occurs for this second probe, no character set conversion is performed (although message reformatting, *e.g.*, changes to MIME structure, may be performed in accordance with the keyword matched on the first probe). If a match does occur it should produce a string of the form:

OUT-CHARSET=*out-char-set*

Here the *out-char-set* specifies the name of the character set to which the *in-char-set* should be converted. Note that both of these character sets must be defined in the character set definition table, `charsets.txt`, located in the PMDF table directory. No conversion will be done if the character sets are not properly defined in this file. This is not usually a problem since this file defines several hundred character sets; most of the character sets in use today are defined in this file. See the description of the `PMDF CHBUILD` (OpenVMS) or `pmdf chbuild` (UNIX and NT) utility in Chapter 29 and Chapter 30 for further information on the `charsets.txt` file.

If all the conditions are met, PMDF will proceed to build the character set mapping and do the conversion. The converted message part will be relabelled with the name of the character set to which it was converted.

6.2.1 Converting DEC-MCS to ISO-8859-1 and Back

Suppose that DEC-MCS is used locally, but this needs to be converted to ISO-8859-1 for use on the Internet. In particular, suppose the connection to the Internet is via a set of TCP channels (including but not limited to `tcp_local`), and `l` and `d` channels are in use locally. The table shown in Example 6–1 brings such conversions about.

Example 6–1 Converting DEC-MCS to and from ISO-8859-1

```
CHARSET-CONVERSION

  IN-CHAN=l;OUT-CHAN=tcp_*;CONVERT                Yes
  IN-CHAN=d;OUT-CHAN=tcp_*;CONVERT                Yes
  IN-CHAN=tcp_*;OUT-CHAN=l;CONVERT                Yes
  IN-CHAN=tcp_*;OUT-CHAN=d;CONVERT                Yes
  IN-CHAN=*;OUT-CHAN=*;CONVERT                    No
  IN-CHAN=l;OUT-CHAN=tcp_*;IN-CHARSET=DEC-MCS     OUT-CHARSET=ISO-8859-1
  IN-CHAN=d;OUT-CHAN=tcp_*;IN-CHARSET=DEC-MCS     OUT-CHARSET=ISO-8859-1
  IN-CHAN=tcp_*;OUT-CHAN=l;IN-CHARSET=ISO-8859-1  OUT-CHARSET=DEC-MCS
  IN-CHAN=tcp_*;OUT-CHAN=d;IN-CHARSET=ISO-8859-1  OUT-CHARSET=DEC-MCS
```

6.2.2 Converting DEC-KANJI to ISO-2022-JP and Back

The table shown in Example 6–2 specifies a conversion between local usage of DEC Kanji and the ISO 2022 based JP code used on the Internet.

Example 6–2 Converting DEC-Kanji to and from ISO-2022-JP

```
CHARSET-CONVERSION

  IN-CHAN=l;OUT-CHAN=l;CONVERT                        No
  IN-CHAN=l;OUT-CHAN=d;CONVERT                        No
  IN-CHAN=d;OUT-CHAN=l;CONVERT                        No
  IN-CHAN=d;OUT-CHAN=d;CONVERT                        No
  IN-CHAN=l;OUT-CHAN=*;CONVERT                        Yes
  IN-CHAN=d;OUT-CHAN=*;CONVERT                        Yes
  IN-CHAN=*;OUT-CHAN=l;CONVERT                        Yes
  IN-CHAN=*;OUT-CHAN=d;CONVERT                        Yes
  IN-CHAN=l;OUT-CHAN=*;IN-CHARSET=DEC-KANJI           OUT-CHARSET=ISO-2022-JP
  IN-CHAN=d;OUT-CHAN=*;IN-CHARSET=DEC-KANJI           OUT-CHARSET=ISO-2022-JP
  IN-CHAN=*;OUT-CHAN=l;IN-CHARSET=ISO-2022-JP         OUT-CHARSET=DEC-KANJI
  IN-CHAN=*;OUT-CHAN=d;IN-CHARSET=ISO-2022-JP         OUT-CHARSET=DEC-KANJI
```

6.3 Message Reformatting

As described above, the CHARSET-CONVERSION mapping is also used to effect the conversion of attachments between MIME and several proprietary mail formats. Use of the Pathworks Mail conversion is described in Section 18.3.

The following sections give examples of some of the other sorts of message reformatting which can be affected with the CHARSET-CONVERSION mapping.

6.3.1 Non-MIME Binary Attachment Conversion

Mail in certain non-standard (non-MIME) formats, *e.g.*, mail in certain proprietary Sun formats or mail from the Microsoft Mail (MSMAIL) SMTP gateway, is automatically converted into MIME format if CHARSET-CONVERSION is enabled for any of the channels involved in handling the message. If you have a tcp_local channel then it is normally the incoming channel for messages from a Microsoft Mail SMTP gateway, and the following will enable the conversion of messages delivered to your local users:

```
CHARSET-CONVERSION

  IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT          Yes
```

You may also want to add entries for channels to other local mail systems. For instance, an entry for the mr_local channel:[3]

```
CHARSET-CONVERSION

  IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT          Yes
  IN-CHAN=tcp_local;OUT-CHAN=mr_local;CONVERT   Yes
```

Alternatively, to cover every channel you can simply specify OUT-CHAN=* instead of OUT-CHAN=l. However, this may bring about an increase in message processing overhead as all messages coming in the tcp_local channel will now be scrutinized instead of just those bound to specific channels.

[3] There is no need to make entries for cc:Mail or Novell MHS channels as they will automatically perform the necessary conversions.

Note: More importantly, such undiscriminated conversions may place your system in the dubious and frowned upon position of converting messages — not necessarily your own site's — which are merely passing through your system, a situation in which you should merely be acting as a transport and not necessarily altering anything beyond the message envelope and related transport information.

To convert MIME into the format Microsoft Mail SMTP gateway understands, use a separate channel in your PMDF configuration for the Microsoft Mail SMTP gateway, *e.g.,* `tcp_msmail`, and put the following in the `mappings.` file:

```
CHARSET-CONVERSION

  IN-CHAN=*;OUT-CHAN=tcp_msmail;CONVERT        RFC1154
```

6.3.2 Relabelling MIME Headers

Some user agents or gateways may emit messages with MIME headers which are less informative than they might be, but which nevertheless contain enough information to construct more precise MIME headers. Although the best solution is to properly configure such user agents or gateways, if they are not under your control, you can instead ask PMDF to try to reconstruct more useful MIME headers.

If the first probe of the `CHARSET-CONVERSION` mapping table yields a "`Yes`" or "`Always`" keyword, then PMDF will check for the presence of a conversions file, (named by the logical `PMDF_CONVERSION_FILE` on OpenVMS or the `PMDF_CONVERSION_FILE` option in the PMDF tailor file on UNIX, hence usually `PMDF_TABLE:conversions.` on OpenVMS, or `/pmdf/table/conversions` on UNIX, or `C:\pmdf\table\conversions` on NT). If a conversions file exists, then PMDF will look in it for an entry with `RELABEL=1` and if it finds such an entry, PMDF will then perform any MIME relabellings specified in the entry. See Section 22.1.3 for complete information on PMDF conversions file entries.

For instance, the combination of a `CHARSET-CONVERSION` table such as

```
CHARSET-CONVERSION

   IN-CHAN=tcp_local;OUT-CHAN=mr_local;CONVERT            Yes
```

and PMDF conversions file entries of

```
out-chan=mr_local; in-type=application; in-subtype=octet-stream;
 in-parameter-name-0=name; in-parameter-value-0=*.ps;
 out-type=application; out-subtype=postscript;
 parameter-copy-0=*; relabel=1

out-chan=mr_local; in-type=application; in-subtype=octet-stream;
 in-disposition=attachment;
 in-dparameter-name-0=filename; in-dparameter-value-0=*.ps;
 out-type=application; out-subtype=postscript;
 out-disposition=attachment; dparameter-copy-0=*; relabel=1

out-chan=mr_local; in-type=application; in-subtype=octet-stream;
 in-parameter-name-0=name; in-parameter-value-0=*.msw;
 out-type=application; out-subtype=msword;
 parameter-copy-0=*; relabel=1
```

```
out-chan=mr_local; in-type=application; in-subtype=octet-stream;
 in-disposition=attachment;
 in-dparameter-name-0=filename; in-dparameter-value-0=*.msw;
 out-type=application; out-subtype=msword;
 out-disposition=attachment; dparameter-copy-0=*; relabel=1
```

will result in messages that arrive on the `tcp_local` channel and are routed to the `mr_local` channel, and that arrive originally with MIME labelling of `application/octet-stream` but have a filename parameter with the extension "`ps`" or "`msw`", being relabelled as `application/postscript` or `application/msword`, respectively. (Note that this more precise labelling is what the original user agent or gateway should have performed itself.) Such a relabelling can be particularly useful in conjunction with a `MIME-CONTENT-TYPES-TO-MR` mapping table, used to convert such resulting MIME types back into appropriate `MRTYPE` tags, which needs precise MIME labelling in order to function optimally; if all content types were left labelled only as `application/octet-stream`, the `MIME-CONTENT-TYPES-TO-MR` mapping table could only, at best, unconditionally convert all such to one sort of `MRTYPE`.

With the above example and `MIME-CONTENT-TYPES-TO-MR` mapping table entries including

```
APPLICATION/POSTSCRIPT            PS
APPLICATION/MSWORD                MW
```

a labelling coming in:

```
Content-type: application/octet-stream; name=stuff.ps
```

would be relabelled as:

```
Content-type: application/postscript
```

and then converted into an `MRTYPE` tag `PS` to let Message Router know to expect PostScript.

Sometimes it is useful to do relabelling in the opposite sort of direction, "downgrading" specific MIME attachment labelling to `application/octet-stream`, the label for generic binary data. In particular, "downgrading" specific MIME labelling is often used in conjunction with the `convert_octet_stream` channel keyword on the `mime_to_x400` channel (PMDF-X400) or `xapi_local` channel (PMDF-MB400) to force *all* binary MIME attachments to be converted to X.400 bodypart 14 format.

For instance, the combination of a `CHARSET-CONVERSION` mapping table such as

```
CHARSET-CONVERSION

   IN-CHAN=*;OUT-CHAN=mime_to_x400*;CONVERT      Yes
```

and PMDF conversions file entries of

```
out-chan=mime_to_x400*; in-type=application; in-subtype=*;
  out-type=application; out-subtype=octet-stream; relabel=1

out-chan=mime_to_x400*; in-type=audio; in-subtype=*;
  out-type=application; out-subtype=octet-stream; relabel=1

out-chan=mime_to_x400*; in-type=image; in-subtype=*;
  out-type=application; out-subtype=octet-stream; relabel=1
```

```
out-chan=mime_to_x400*; in-type=video; in-subtype=*;
  out-type=application; out-subtype=octet-stream; relabel=1
```

will result in "downgrading" various specific MIME attachment labelling to the generic `application/octet-stream` labelling (so that `convert_octet_stream` will apply) for all messages going to `mime_to_x400*` channels.

6.3.3 MacMIME Format Conversions

Macintosh files have two parts, a resource fork which contains Macintosh specific information, and a data fork which contains data usable on other platforms. This introduces an additional complexity when transporting Macintosh files, as there are four different formats in common use for transporting the Macintosh file parts.[5] Three of the formats, Applesingle, Binhex, and Macbinary, consist of the Macintosh resource fork and Macintosh data fork encoded together in one piece. The fourth format, Appledouble, is a multipart format with the resource fork and data fork in separate parts. Appledouble is hence the format most likely to be useful on non-Macintosh platforms, as in this case the resource fork part may be ignored and the data fork part is available for use by non-Macintosh applications. But the other formats may be useful when sending specifically to Macintoshes.

PMDF can convert between these various Macintosh formats. The `CHARSET-CONVERSION` keywords `Appledouble` `Applesingle`, `Binhex`, or `Macbinary` tell PMDF to convert other MacMIME structured parts to a MIME structure of `multipart/appledouble`, `application/applefile`, `application/mac-binhex40`, or `application/macbinary`, respectively. Further the `Binhex` or `Macbinary` keywords also request conversion to the specified format of non-MacMIME format parts that do nevertheless contain `X-MAC-TYPE` and `X-MAC-CREATOR` parameters on the MIME `Content-type:` header. The `CHARSET-CONVERSION` keyword `Block` tells PMDF to extract just the data fork from MacMIME format parts, discarding the resource fork; (since this loses information, use of `Appledouble` instead is generally preferable).

For instance, the following `CHARSET-CONVERSION` table would tell PMDF to convert to Appledouble format when delivering to the VMS MAIL mailbox or a GroupWise postoffice, and to convert to Macbinary format when delivering to the Message Router channel:

```
CHARSET-CONVERSION

  IN-CHAN=*;OUT-CHAN=l;CONVERT           Appledouble
  IN-CHAN=*;OUT-CHAN=wpo_local;CONVERT   Appledouble
  IN-CHAN=*;OUT-CHAN=mr_local;CONVERT    Macbinary
```

The conversion to Appledouble format would only be applied to parts already in one of the MacMIME formats. The conversion to Macbinary format would only be applied to parts already in one of the MacMIME formats, or non-MacMIME parts which included `X-MAC-TYPE` and `X-MAC-CREATOR` parameters on the MIME `Content-type:` header.

[5] See RFC 1740 (MacMIME) and RFC 1741 (Binhex).

When doing conversion to Appledouble or Block format, the MAC-TO-MIME-CONTENT-TYPES mapping table may be used to indicate what specific MIME label to put on the data fork of the Appledouble part, or the Block part, depending on what the Macintosh creator and Macintosh type information in the original Macintosh file were. Probes for this table have the form *format*|*type*|*creator*|*filename* where *format* is one of SINGLE, BINHEX or MACBINARY, where *type* and *creator* are the Macintosh type and Macintosh creator information in hex, respectively, and where *filename* is the file name. For instance, to convert to Appledouble when sending to the l channel and when doing so to use specific MIME labels for any MS Word or PostScript documents converted from MACBINARY or BINHEX parts, appropriate tables might be:

```
CHARSET-CONVERSION

  IN-CHAN=*;OUT-CHAN=l;CONVERT     Appledouble

MAC-TO-MIME-CONTENT-TYPES

! PostScript
   MACBINARY|45505346|76677264|*     APPLICATION/POSTSCRIPT$Y
   BINHEX|45505346|76677264|*        APPLICATION/POSTSCRIPT$Y
! Microsoft Word
   MACBINARY|5744424E|4D535744|*     APPLICATION/MSWORD$Y
   BINHEX|5744424E|4D535744|*        APPLICATION/MSWORD$Y
```

Note that the template (right hand side) of the mapping entry must have the $Y flag set in order for the specified labelling to be performed. Sample entries for additional types of attachments may be found in the file mac_mappings.sample in the PMDF table directory.

If you want to convert non-MacMIME format parts to Binhex or Macbinary format, such parts need to have X-MAC-TYPE and X-MAC-CREATOR MIME Content-type: parameter values provided. Note that MIME relabelling can be used to force such parameters onto parts that would not otherwise have them; see Section 6.3.2 for a discussion of MIME relabelling.

6.4 Service Conversions

PMDF's conversion service facility may be used to process with site-supplied procedures a message so as to produce a new form of the message. Unlike either the sorts of CHARSET-CONVERSION operations discussed above or the conversion channel, which operate on the content of individual MIME message parts, conversion services operate on entire MIME message parts (MIME headers and content) as well as entire MIME messages. Also, unlike other CHARSET-CONVERSION operations or conversion channel operations, conversion services are expected to do their own MIME disassembly, decoding, re-encoding, and reassembly.

Like other CHARSET-CONVERSION operations, conversion services are enabled through the CHARSET-CONVERSION mapping table. If the first probe of the CHARSET-CONVERSION

mapping table yields a "Yes" or "Always" keyword, then PMDF will check for the presence of a PMDF conversions file. [6] If a conversions file exists, then PMDF will look in it for an entry specifying a SERVICE-COMMAND, and if it finds such an entry, execute it. The conversions file entries should have the form

```
in-chan=channel-pattern;
  in-type=type-pattern; in-subtype=subtype-pattern;
  service-command=command
```

Of key interest is the *command* string. This is the command which should be executed to perform a service conversion (*e.g.,* invoke a document converter). The command must process an input file containing the message text to be serviced and produce as output a file containing the new message text. On OpenVMS, the command must exit with an odd-valued status code if successful and an even-valued status code if unsuccessful. On UNIX, the command must exit with a 0 if successful and a non-zero value otherwise.

For instance, the combination of a CHARSET-CONVERSION table such as

```
CHARSET-CONVERSION

  IN-CHAN=bsout_*;OUT-CHAN=*;CONVERT     Yes
```

and a PMDF conversions file entry on OpenVMS of

```
in-chan=bsout_*; in-type=*; in-subtype=*;
  service-command="@PMDF_COM:COMPRESS.COM COMPRESS 'INPUT_FILE' 'OUTPUT_FILE'"
```

or on UNIX of

```
in-chan=bsout_*; in-type=*; in-subtype=*;
  service-command="/pmdf/bin/compress.sh compress $INPUT_FILE $OUTPUT_FILE"
```

or on NT of

```
in-chan=bsout_*; in-type=*; in-subtype=*;
  service-command="c:\pmdf\bin\compress.exe %INPUT_FILE% %OUTPUT_FILE%"
```

will result in all messages coming from a bsout_* channel being compressed.

DCL symbols (OpenVMS) or environment variables (UNIX and NT) are used to pass the names of the input and output files as well as the name of a file containing the list of the message's envelope recipient addresses. The names of these environment variables are:

Variable	Usage
INPUT_FILE	Name of the input file to process
OUTPUT_FILE	Name of the output file to produce
INFO_FILE	Name of the file containing envelope recipient addresses

The values of these three environment variables may be substituted into the command line by using standard command line substitution: *i.e.*, preceding and following the

[6] The conversions file is located via the PMDF_CONVERSION_FILE logical (OpenVMS) or PMDF tailor file option (UNIX), or Registry entry (NT), and is usually PMDF_TABLE:conversions. (OpenVMS) or /pmdf/table/conversions (UNIX) or C:\pmdf\table\conversions (NT).

variable's name with an apostrophe on OpenVMS, preceding the variable's name with a dollar character on UNIX, or preceding and following the variable's name with a percent sign on NT. For example, when INPUT_FILE and OUTPUT_FILE have the values a.in and a.out, then the following declaration on OpenVMS,

```
in-chan=bsout_*; in-type=*; in-subtype=*;
  service-command="@PMDF_COM:CONVERT.COM 'INPUT_FILE' 'OUTPUT_FILE'"
```

executes the command

```
@PMDF_COM:CONVERT.COM A.IN A.OUT
```

On UNIX, the declaration

```
in-chan=bsout_*; in-type=*; in-subtype=*;
  service-command="/pmdf/bin/convert.sh $INPUT_FILE $OUTPUT_FILE"
```

executes the command

```
/pmdf/bin/convert.sh a.in a.out
```

On NT, the declaration

```
in-chan-bsout_*; in-type=*; in-subtype=*;
  service-command="c:\pmdf\bin\convert.exe %INPUT_FILE% %OUTPUT_FILE%"
```

executes the command

```
c:\pmdf\bin\convert.exe a.in a.out
```

6.5 Complex Conversions

PMDF's native conversion facilities are fairly limited, so the ability to call external converters is crucial.

For complex conversions of message attachments using external, third-party programs and site-supplied procedures, such as document convertors, see Chapter 22. Such complex conversions can be performed by the script and conversion channels.

Invoking the conversion channel is controlled by the CONVERSIONS mapping table, and invoking the script channel is controlled by the SCRIPT mapping table. The conversion channel or the script channel is then run to execute a site-specified external conversion procedure.

The PMDF conversions file, discussed in Section 22.1.3, is used to specify the details of external CONVERSIONS table triggered conversions and to specify the details of some internal CHARSET-CONVERSION table triggered conversions.

Note: If you have both a CONVERSIONS mapping table and a CHARSET-CONVERSION mapping table, the CONVERSIONS mapping table takes precedence. In this case, all entries with RELABEL=1 in the conversions file are skipped. A conversions file entry can be changed from being associated with the CHARSET-CONVERSION mapping table to being associated with the CONVERSIONS mapping table by replacing the RELABEL=1

parameter with the COMMAND parameter. The command specified could be as simple as copying the input file to the output file.

7 The PMDF Option File

PMDF uses an option file to provide a means of overriding the default values of various parameters that apply to PMDF as a whole. Various PMDF channels also have their own channel-level option files. This (global) PMDF option file has the same format but is otherwise distinct — it applies to PMDF as a whole and not to any specific channel.

A variety of configuration options are controlled by options in the PMDF option file. In particular, the option file is used to establish sizes of the various tables into which the configuration and alias files are read.

7.1 Locating and Loading the Option File

On OpenVMS systems, the option file is referenced via the PMDF_OPTION_FILE logical name. By default, this points to the file `PMDF_TABLE:option.dat`. On UNIX systems, the option file is the file specified with the PMDF_OPTION_FILE option in the PMDF tailor file.[1] By default, this is file `/pmdf/table/option.dat`. On NT systems, the option file is the file specified with the PMDF_OPTION_FILE PMDF Tailor NT Registry entry. Typically this points to the file `C:\pmdf\table\option.dat`.

Each time a PMDF program begins running, this file is read and loaded into memory. This overhead can be avoided by compiling your PMDF configuration, in which case the contents of the option file will be incorporated into the compiled configuration. The disadvantage to this, however, is that it means that the configuration must be recompiled and reinstalled whenever a change is made to the option file. See Chapter 8 for details on compiling your configuration.

The PMDF option file should be world readable.

7.2 Option File Format

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

option=value

value can be either a string or an integer, depending on the option's requirements. If the option accepts an integer value a base can be specified using notation of the form *b%v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

[1] The PMDF tailor file is `/etc/pmdf_tailor`.

Comments are allowed. Any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored in any option file.

7.3 Available Options

Table 7–1 lists the available options. Further descriptions of the options can be found in the subsections below.

Table 7–1 PMDF Global Option File Options

	Option	Section	Usage
	ACCESS_ERRORS	7.3.4	Control the information issued in certain error messages
	ACCESS_ORCPT	7.3.12	Include the original recipient information in SEND_ACCESS and ORIG_SEND_ACCESS mapping table probes.
	ALIAS_DOMAINS	7.3.1	Control the format of alias file and alias database lookups
	ALIAS_HASH_SIZE	7.3.9	Set the number of aliases allowed in the alias file
	ALIAS_MEMBER_SIZE	7.3.9	Set the number of alias expansions allowed in the alias file
	BLOCK_LIMIT	7.3.5	Limit the size of messages allowed through PMDF
	BLOCK_SIZE	7.3.5	Set the size of PMDF "blocks"
	BOUNCE_BLOCK_LIMIT	7.3.5	Limit the amount of original message content included in bounce messages
	CHANNEL_TABLE_SIZE	7.3.9	Set the number of channels allowed in the PMDF configuration
	CIRCUITCHECK_COMPLETED_BINS	7.3.6	Specify the bins for the message circuit check message counters
	CIRCUITCHECK_PATHS_SIZE	7.3.9	Number of circuit check paths (entries) to allow in the circuit check configuration file
	COMMENT_CHARS	7.3.8	Set the "comment" character(s) in PMDF configuration files
	CONTENT_RETURN_BLOCK_LIMIT	7.3.5	Force NOTARY non-return of content flag for messages over the specified size
	CONVERSION_SIZE	7.3.9	Set the number of entries allowed in the conversion file
†	DELIVERY_RECEIPT_OFF	7.3.11	Specify the RFC 822 comment string for disabling delivery receipt requests
†	DELIVERY_RECEIPT_ON	7.3.11	Specify the RFC 822 comment string for requesting a delivery receipt
	DEQUEUE_DEBUG	7.3.10	Enable debugging of message dequeue operations
	DISABLE_DELIVERY_RECEIPT	7.3.4	Disable the generation of all successful delivery receipts.
†	DIS_NESTING	7.3.11	Control the level of nesting allowed for VMS MAIL @DIS distribution lists
	DOMAIN_HASH_SIZE	7.3.9	Set the number of rewrite rules allowed

†Available only on OpenVMS

Table 7–1 (Cont.) PMDF Global Option File Options

	Option	Section	Usage
	EXPROUTE_FORWARD	7.3.1	Control whether the `exproute` keyword affects forward pointing headers
	FILTER_DISCARD	7.3.3	Control whether messages discarded by a mailbox filter are immediately deleted, or instead routed to the filter_discard channel for delayed deletion
†	FORM_NAMES	7.3.11	List the names of pop-up form images
‡	FSYNC	7.3.8	Do an fsync upon file close
	HELD_SNDOPR	7.3.7	Send operator or syslog messages when messages are HELD
	HISTORY_TO_RETURN	7.3.4	Control the amount of delivery attempt history included in bounced messages
	HOST_HASH_SIZE	7.3.9	Set the number of channel host names
	ID_DOMAIN	7.3.1	Set the domain name used in constructing message IDs
	IMPROUTE_FORWARD	7.3.1	Control the effect of the `improute` keyword on forward pointing headers
	INCLUDE_CONNECTIONINFO	7.3.12	Include the transport and application connection information in various mapping table probes.
	LDAP_HOST	7.3.2	Host to which to connect for LDAP queries
	LDAP_PASSWORD	7.3.2	The password to use when binding for LDAP queries
	LDAP_PORT	7.3.2	Port to which to connect for LDAP queries
	LDAP_TIMEOUT	7.3.2	Timeout value for LDAP queries
	LDAP_TLS_MODE	7.3.2	Whether to use TLS for LDAP queries
	LDAP_USERNAME	7.3.2	The DN under which to bind for LDAP queries
	LINE_LIMIT	7.3.5	Limit the size of messages allowed through PMDF
	LINES_TO_RETURN	7.3.4	Lines included when returning samples of message content (as in warning messages)
†	LOG_ALQ	7.3.6	Specify the default allocation quantity for the PMDF log file
	LOG_CONNECTION	7.3.6	Include connection information in log entries
	LOG_DELAY_BINS	7.3.6	Specify the bins for delivery delay range counters
†	LOG_DEQ	7.3.6	Specify the default extend quantity for the PMDF log file
	LOG_FILENAME	7.3.6	Include message file names in PMDF log entries
	LOG_FORMAT	7.3.6	Control the format of the PMDF log file
	LOG_HEADER	7.3.6	Include message headers in PMDF log entries
	LOG_LOCAL	7.3.6	Include the local domain name on “bare user name” addresses in PMDF log entries
	LOG_MESSAGE_ID	7.3.6	Include message IDs in PMDF log entries
	LOG_MESSAGES_SYSLOG	7.3.6	Send PMDF log file entries to syslog (UNIX) or event log (NT)
†	LOG_NODE	7.3.6	Include the node name for an enqueueing process in PMDF log entries
†	LOG_NOTARY	7.3.6	Include a NOTARY (delivery receipt) flags indicator in PMDF log entries

†Available only on OpenVMS

‡Available only on UNIX and NT

Table 7–1 (Cont.) PMDF Global Option File Options

	Option	Section	Usage
	LOG_PROCESS	7.3.6	Include enqueuing process ID in PMDF log entries
	LOG_SENSITIVITY	7.3.6	Include message's sensitivity value in log entries
	LOG_SIZE_BINS	7.3.6	Specify the bins for message size range counters
	LOG_SNDOPR	7.3.6	Send an operator or syslog message if PMDF's logging facilities encounter a difficulty
	LOG_USERNAME	7.3.6	Include the username for an enqueuing process in PMDF log entries
†	MAIL_DELIVERY_FILENAME	7.3.11	Specify the file name used by DELIVER
	MAIL_OFF	7.3.1	Specify the comment string that disables mail delivery for list addresses
	MAP_NAMES_SIZE	7.3.9	Set the number of mapping tables
	MAX_ALIAS_LEVELS	7.3.1	Set the level of alias nesting allowed
	MAX_FILEINTOS	7.3.3	Maximum number of files that can be specified by a mailbox filter's fileinto operator
	MAX_FORWARDS	7.3.3	Maximum number of forwarding addresses that can be specified by a mailbox filter's forward operator
	MAX_HEADER_BLOCK_USE	7.3.1	Fine tune message fragmentation
	MAX_HEADER_LINE_USE	7.3.1	Fine tune message fragmentation
	MAX_INLINE_DIR_LEVELS	7.3.1	Set the level of inline directory channel lookup nesting allowed
	MAX_INTERNAL_BLOCKS	7.3.5	Specify size of messages beyond which to buffer to temporary files
	MAX_LIST_SIZE	7.3.3	Maximum number of entries that can be in a mailbox filter's list
	MAX_LOCAL_RECEIVED_LINES	7.3.7	Occurrences of the local host name in Received: headers after which a message will be HELD
	MAX_MIME_LEVELS	7.3.5	Degree to look inside MIME messages during processing
	MAX_MIME_PARTS	7.3.5	Number of parts to look at when processing MIME messages
	MAX_MR_RECEIVED_LINES	7.3.7	Number of MR-Received: headers after which a message will be HELD
	MAX_RECEIVED_LINES	7.3.7	Number of Received: headers after which a message will be HELD
	MAX_TOTAL_RECEIVED_LINES	7.3.7	Number of Received:, MR-Received: or X400-Received: headers after which a message will be HELD
	MAX_URLS	7.3.2	Maximum number of URLs that can be active (nesting of references)
	MAX_X400_RECEIVED_LINES	7.3.7	Number of X400-Received: headers after which a message will be HELD
†	MISSING_ADDRESS	7.3.11	VMS MAIL From: address to substitute for a missing address
	MISSING_RECIPIENT_POLICY	7.3.1	Legalize messages that lack any recipient headers
	MP_SIGNED_MODE	7.3.11	Control PMDF's support of multipart/signed messages

†Available only on OpenVMS

Table 7–1 (Cont.) PMDF Global Option File Options

	Option	Section	Usage
†	MULTINET_MM_EXCLUSIVE	7.3.11	Control whether delivery is to VMS MAIL or MultiNet MM
†	NAME_TABLE_NAME	7.3.1	Control whether logical name tables can be used for aliasing
	NORMAL_BLOCK_LIMIT	7.3.5	Maximum size of message to treat as being of normal or higher priority
	NON_URGENT_BLOCK_LIMIT	7.3.5	Maximum size of message to treat as being of non-urgent priority
	POST_DEBUG	7.3.10	Enable debugging of PMDF periodic delivery job operations
†	READ_RECEIPT_OFF	7.3.11	Specify the RFC 822 comment string for disabling read receipt requests
†	READ_RECEIPT_ON	7.3.11	Specify the RFC 822 comment string for requesting a read receipt
	RECEIVED_DOMAIN	7.3.1	Specify the domain name (identifying the system itself) to use in constructing Received: headers
	RETURN_ADDRESS	7.3.4	Set the return address for the local postmaster
	RETURN_DEBUG	7.3.10	Enable debugging of PMDF periodic return job operations
	RETURN_DELIVERY_HISTORY	7.3.4	Control whether delivery attempt history is included in returned messages
†	RETURN_DELTA	7.3.4	Set the offset from midnight, or delta time between runs, of the execution of the PMDF periodic return job
	RETURN_ENVELOPE	7.3.4	Control use of empty return address in notification messages
	RETURN_PERSONAL	7.3.4	Set the personal name for the postmaster
	RETURN_UNITS	7.3.4	Control whether the PMDF periodic return job runs on a daily or hourly schedule
	REVERSE_ENVELOPE	7.3.1	Control application of address reversal to envelope addresses
	REVERSE_URL	7.3.2	URL for doing address reversal
†	SAFE_TCL_MODE	7.3.11	Control PMDF MAIL's support of Safe-Tcl message parts
	SEPARATE_CONNECTION_LOG	7.3.6	Write connection log entries to a separate file than message log entries
	SNDOPR_PRIORITY	7.3.6	Set the priority of operator broadcast or the syslog level of syslog messages
	STRING_POOL_SIZE	7.3.9	Set the number of strings allowed for general PMDF configuration use
	SUPPRESS_CONTENT_DISP	7.3.1	Suppress the generation of Content-disposition headers
	URGENT_BLOCK_LIMIT	7.3.5	Maximum size of message to treat as being of urgent priority
	USE_ALIAS_DATABASE	7.3.1	Control use of the alias database
	USE_DOMAIN_DATABASE	7.3.1	Control use of the domain database
	USE_ERRORS_TO	7.3.4	Control use of Errors-to: information when returning messages
	USE_FORWARD_DATABASE	7.3.1	Control use of the forward database
†	USE_MAIL_DELIVERY	7.3.11	Control whether users can use DELIVER

†Available only on OpenVMS

Table 7–1 (Cont.) PMDF Global Option File Options

	Option	Section	Usage
	USE_PERSONAL_ALIASES	7.3.1	Control use of personal alias databases
	USE_REVERSE_DATABASE	7.3.1	Control use and format of the REVERSE mapping and reverse database
	USE_WARNINGS_TO	7.3.4	Control use of Warnings-to: information when returning messages
†	VMS_MAIL_EXCLUSIVE	7.3.11	Control whether delivery is to VMS MAIL or MultiNet MM
	WILD_POOL_SIZE	7.3.9	Set the total number of wildcards allowed in mappings patterns

†Available only on OpenVMS

7.3.1 Addresses, Aliases, Headers, and Rewriting Options

The options described in this section affect and modify various aspects of PMDF address, alias, and rewriting handling, and the information placed in certain sorts of headers.

ALIAS_DOMAINS (integer)
This option takes a bit encoded integer argument controlling the format of alias file and alias database lookups. The default value is 1, meaning that alias file and alias database lookups probe with only the local part (mailbox portion) of the address. Note that for addresses matching the local channel, such a probe is made even if bit 0 (value 1) is not set. Setting bit 1 (value 2) causes a probe to be made using the entire address (including the domain name). Setting bit 2 (value 4) causes a wildcard * probe to be made, (akin to the sort of wildcard * probe made when doing a directory channel `crdb` lookup). If all bits are set, *i.e.*, ALIAS_DOMAINS=7, then the order of the probes is to first probe with the entire address (the most specific check), next probe with a wildcard * local part plus the domain name, and finally probe with just the local part.

Bit	Value	Usage
0	1	Look up *localpart*. Clearing this bit disables the lookup of local parts only for channels other than the local channel; for the local channel, local parts are always looked up.
1	2	Look up *localpart@domainname*.
2	4	Try an * lookup if no exact match is found.

Bit 0 is the least significant bit.

Note that by default only addresses rewritten to the local channel are checked against the alias file and alias database. However, via use of the `aliaslocal` channel keyword it is possible to cause addresses matching other channels to be checked against the alias file and alias database.

EXPROUTE_FORWARD (integer 0 or 1)

This option controls the application of the `exproute` channel keyword to forward-pointing (To:, Cc:, and Bcc: lines) addresses in the message header. A value of 1 is the default and specifies that `exproute` should affect forward-pointing header addresses. A value of 0 disables the action of the `exproute` keyword on forward-pointing addresses.

ID_DOMAIN (string)

The ID_DOMAIN specifies the domain name to use when constructing message IDs. By default, the official host name of the local channel is used.

IMPROUTE_FORWARD (integer 0 or 1)

This option controls the application of the `improute` channel keyword to forward-pointing (To:, Cc:, and Bcc: lines) addresses in the message header. A value of 1 is the default and specifies that `improute` should affect forward-pointing header addresses. A value of 0 disables the action of the `improute` keyword on forward-pointing addresses.

MAIL_OFF (string)

Specify the comment string that disables mail delivery for list addresses. The default is `NOMAIL`.

MAX_ALIAS_LEVELS (integer)

The MAX_ALIAS_LEVELS option controls the degree of indirection allowed in aliases, that is, how deeply aliases can be nested, with one alias referring to another alias, *etc*. The default value is 10.

MAX_INLINE_DIR_LEVELS (integer)

The MAX_INLINE_DIR_LEVELS option controls the degree of indirection allowed in directory channel inline lookups, that is, how deeply directory channel aliases can be nested, with one directory channel entry referring to another directory channel entry, *etc*. The default value is 10. Note that this option only applies when the `inline` channel keyword is used on the directory channel.

MISSING_RECIPIENT_POLICY (integer)

According to RFC 822, messages are required to contain at least one recipient header: a `To:`, `Cc:`, or `Bcc:` header. This RFC states that a message without any such headers is illegal. This requirement has been relaxed in the updated RFC 2822 standard: such messages are no longer illegal. However, some remote systems that conform to RFC 822 will not accept these messages. In many cases, it can be useful to have PMDF modify the message to include at least one recipient header.

The MISSING_RECIPIENT_POLICY option takes an integer value specifying what approach to use for such messages; the default value, if the option is not explicitly present, is 1, meaning that no action is taken.

Value	Action
1	Pass the message through unchanged
2	Place envelope To: recipients in a To: header
3	Place all envelope To: recipients in a single Bcc: header
4	Generate an empty group construct To: header (i.e. "To: Recipients not specified: ;")
5	Generate a blank Bcc: header
6	Reject the message

Note that the The `missingrecipientpolicy` channel keyword, discussed in Section 2.3.4.48, can be used to set per-channel controls for this sort of behavior.

NAME_TABLE_NAME (string; OpenVMS only)
The NAME_TABLE_NAME option specifies the name of a logical name table to be searched for address aliases by PMDF. This table name can itself be a logical name (in the process or system directory) which specifies one or more tables to search. This option has no default; if it is not specified logical name tables are not searched for aliases.

RECEIVED_DOMAIN (string)
The RECEIVED_DOMAIN option sets the domain name to use when constructing Received: headers. By default, the offical host name of the local channel is used.

REVERSE_ENVELOPE (0 or 1)
The REVERSE_ENVELOPE option controls whether or not PMDF applies address reversal to envelope From: addresses as well as header addresses. This option will have no effect if the USE_REVERSE_DATABASE option is set to 0 or if neither the reverse database nor a REVERSE mapping exist. The default is 1, which means that PMDF will attempt to apply any address reversal to envelope From: addresses. A value of 0 will disable this use of the address reversal database and REVERSE mapping.

SUPPRESS_CONTENT_DISP (integer 0, 1, or 2)
This option suppresses the addition of a `Content-disposition` header to a message or message part when PMDF parses out and re-assembles the MIME parts of a message. PMDF can be configured to do this in several ways, for example, by using the `inner` or `innertrim` channel keywords, or by using the CHARSET-CONVERSION mapping table.

Normally, when parsing the MIME part headers, if PMDF finds a `name` parameter in the `Content-type` header, it will add a `Content-disposition` header with a `filename` parameter. This appears to cause problems with messages generated by Outlook Calendar, with message parts that are of type `text/calendar` and do not have a Content-disposition header. The generation of this header by PMDF can be suppressed by specifing the SUPPRESS_CONTENT_DISP option. A value of 1 always suppresses the generation of a Content-disposition header, and a value of 2 suppresses the generation of the Content-disposition header only for `text/calendar` message parts.

USE_ALIAS_DATABASE (integer 0, 1, or 2)
The USE_ALIAS_DATABASE option controls whether and how PMDF makes use of the alias database as a source of system aliases for local addresses. A value of 0 disables use of the alias database. A value of 1, the default, causes PMDF to check the database if it exists. A value of 2 requires use the alias database. With this setting, if the database does not exist or is inaccessable for any other reason, all messages will be rejected with a temporary error.

USE_DOMAIN_DATABASE (0 or 1)
The USE_DOMAIN_DATABASE option controls whether or not PMDF makes use of the domain database as a source of rewrite rules. The default is 1, which means that PMDF will check the database if it exists. A value of 0 will disable this use of the domain database.

USE_FORWARD_DATABASE (integer)
The USE_FORWARD_DATABASE controls whether or not PMDF makes use of the forward database. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Bit	Value	Usage
0	1	When set, the forward database is used.
3	8	When set, channel-level granularity is used with the forward database entries. Forward database entries' left hand sides must have the form (note the vertical bars, |) *source-channel*|*from-address*|*to-address*
4	16	When set, channel-level granularity is used with the FORWARD mapping. FORWARD mapping entries' patterns (left hand sides) must have the form (note the vertical bars, |) *source-channel*|*from-address*|*to-address*

Bit 0 is the least significant bit.

The default value for USE_FORWARD_DATABASE is 0, which means that PMDF will not use the forward database at all. Note that a FORWARD mapping, if present, is always consulted.

USE_PERSONAL_ALIASES (0 or 1)

The USE_PERSONAL_ALIASES option controls whether or not PMDF makes use of personal alias databases as a source of aliases for local addresses. The default is 1, which means that PMDF will check such databases, if they exist. A value of 0 will disable personal aliases and make them unavailable to all users.

USE_REVERSE_DATABASE (0-511)

The USE_REVERSE_DATABASE option controls whether or not PMDF makes use of the address reversal database and REVERSE mapping as a source of substitution addresses. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Bit	Value	Usage
0	1	When set, address reversal is applied to addresses after they have been rewritten by the PMDF address rewriting process.
1	2	When set, address reversal is applied before addresses have had PMDF address rewriting applied to them.
2	4	When set, address reversal will be applied to all addresses, not just to backwards-pointing addresses.
3	8	When set, channel-level granularity is used with the REVERSE mapping. REVERSE mapping table (pattern) entries must have the form (note the vertical bars, |) *source-channel*|*destination-channel*|*address*
4	16	When set, channel-level granularity is used with address reversal database entries. Reversal database entries' left hand sides must have the form (note the vertical bars, |) *source-channel*|*destination-channel*|*address*
5	32	Apply REVERSE mapping even if a reverse database entry has already matched.
6	64	Apply address reversal to message ids.

Bit 0 is the least significant bit.

Bit	Value	Usage
7	128	When set, this modifies the effect of bit 4 (channel-level granularity of address reversal database entries); when this bit is also set, the address reversal database entries take the form (note the vertical bars, |) *destination-channel*|*address*
8	256	When set, this modifies the effect of bit 3 (channel-level granularity of REVERSE mapping table entries); when this bit is also set, the REVERSE mapping table entries take the form (note the vertical bars, |) *destination-channel*|*address*

Bit 0 is the least significant bit.

The default value for USE_REVERSE_DATABASE is 5, which means that PMDF will reverse Envelope From: addresses and both backwards and forwards pointing addresses after they have passed through the normal address rewriting process. Simple address strings are presented to both the REVERSE mapping and the reverse database. Note that a value of 0 disables the use of the address reversal completely.

Note that the default of 5 represents a change from earlier versions of PMDF in which this option had a default value of 1 (reverse only backwards pointing addresses).

7.3.2 LDAP and URL Lookup Options

This section lists options affecting LDAP and URL lookups.

LDAP_HOST (host name)
Specify the default host to which to connect when making LDAP queries.

LDAP_PASSWORD (string)
The password to use when binding for LDAP queries.

LDAP_PORT (integer)
Specify the port to which to connect when making LDAP queries. The default value is 389, the standard LDAP port number.

LDAP_TIMEOUT (integer)
Control how long to wait (in hundredths of seconds) before timing out on an LDAP query. The default value is 200.

LDAP_TLS_MODE (1 or 2)
Control whether TLS is used for LDAP queries. The default if the option is not specified is to not use TLS. A value of 1 tells PMDF to try to use TLS to look up the alias in LDAP, but continue without it if TLS is not available. A value of 2 tells PMDF to require TLS. Note that in order to use TLS, your LDAP server must be configured to do TLS on its end.

You may need to have the Certificate Authority (CA) certificate to be used by LDAP on your PMDF system. If so, the CA certificate should be placed in the file `pmdf_table:ldap-cacert.pem`.

LDAP_USERNAME (distinguished-name)
The DN under which to bind for LDAP queries.

MAX_URLS (integer)
Maximum number of URLs that can be active when reiteratively performing URL lookups; that is, this is the maximum degree of nesting of URL references. The default value is 5.

REVERSE_URL (URL)
URL to query for address reversal. Standard LDAP URL syntax is used, except omitting the LDAP server and port which are instead specified via the LDAP_HOST and LDAP_PORT options. Also, certain substitution sequences are available, as shown in Table 3–1.

7.3.3 Mailbox Filter Options

This section lists options affecting mailbox filters.

FILTER_DISCARD (1, 2, or 3)
This option controls whether mailbox filter discard actions cause such discarded messages to go to the bitbucket channel (*i.e.*, be immediately discarded), or cause such messages to go to the filter_discard channel (which will leave them around for a short period before discarding them). The default is FILTER_DISCARD=1, meaning that messages discarded by a mailbox filter are immediately discarded. Setting FILTER_DISCARD=2 causes discarded messages to instead be routed to the filter_discard channel; see Section 16.2.5. Setting FILTER_DISCARD=3 causes messages to be routed to the filter_discard channel as with FILTER_DISCARD=2, however these messages are not added to the PMDF queue cache database, improving performance.

MAX_FILEINTOS (integer)
This option specifies the maximum number of folders that can be specified by a mailbox filter's fileinto operator. The default is 10.

MAX_FORWARDS (integer)
This option specifies the maximum number of forwarding addresses that can be specified by a mailbox filter's redirect operator. The default is 32.

MAX_LIST_SIZE (integer)
This option specifies the maximum number of entries that can be in a mailbox filter's list type. The default is 64.

7.3.4 Notification Messages and Jobs Options

This section lists options affecting notification messages and the PMDF periodic return job. See also the BOUNCE_BLOCK_LIMIT and CONTENT_RETURN_BLOCK_LIMIT options, discussed in Section 7.3.5.

ACCESS_ERRORS (integer 0 or 1)

PMDF provides facilities to restrict access to channels on the basis of the NETMBX privilege or rightslist identifiers on OpenVMS, or on the basis of group ids on UNIX. If ACCESS_ERRORS is set to 0 (the default), when an address causes an access failure PMDF will report it as an "illegal host or domain" error. This is the same error that would occur if the address was simply illegal. Although confusing, this usage nevertheless provides an important element of security in circumstances where information about restricted channels should not be revealed. Setting ACCESS_ERRORS to 1 will override this default and provide a more descriptive error.

DISABLE_DELIVERY_RECEIPT (integer 0 or 1)

Setting DISABLE_DELIVERY_RECEIPT to 1 disables the generation of all successful delivery receipts. The default is 0.

HISTORY_TO_RETURN (1-200)

The HISTORY_TO_RETURN option controls how many delivery attempt history records are included in returned messages. The delivery history provides some indication of how many delivery attempts were made and in some cases indicates the reason the delivery attempts failed. The default value for this option is 20.

LINES_TO_RETURN (integer)

The LINES_TO_RETURN option controls how many lines of message content PMDF includes when generating a notification message for which it is appropriate to return only a sample of the contents. The default is 20. If the value 0 is specified, only headers are included.

Note that this option is irrelevant when generating a NOTARY bounce message, where either the full content or merely headers are included, according to the choice specified during the initial submission of the message. So in practice, this option is mostly only relevant to the warning messages the PMDF return job sends about messages awaiting further delivery retries in the PMDF queue area.

RETURN_ADDRESS (string)

The RETURN_ADDRESS option sets the return address for the local Postmaster. The local Postmaster's address is `postmaster@`*`localhost`* by default, but it can be overriden with the address of your choice. Care should be taken in the selection of this address — an illegal selection can cause rapid message looping and pile-ups of huge numbers of spurious error messages.

RETURN_DELIVERY_HISTORY (0 or 1)

This flag controls whether or not a history of delivery attempts is included in returned messages. The delivery history provides some indication of how many delivery attempts were made and in some cases indicates the reason the delivery attempts failed. A value of 1 enables the inclusion of this information and is the default. A value of 0 disables return of delivery history information. The HISTORY_TO_RETURN option controls how much history information is actually returned.

RETURN_DELTA (+01:00:00 or +00:30:00; OpenVMS only)

On OpenVMS systems, the RETURN_DELTA option controls when the message return system runs. When RETURN_UNITS is set to 0, RETURN_DELTA should be set to the time after midnight when the daily job should start. The default is 30 minutes after midnight. When RETURN_UNITS is set to 1, RETURN_DELTA should be set to the interval between runs. In this case the default is one hour. In either case the value is given as a standard VMS delta time.

RETURN_ENVELOPE (integer)

The RETURN_ENVELOPE option takes a single integer value, which is interpreted as a set of bit flags. Bit 0 (value = 1) controls whether or not return notifications generated by PMDF are written with a blank envelope address or with the address of the local postmaster. Setting the bit forces the use of the local postmaster address, clearing the bit forces the use of a blank address. Note that the use of a blank address is mandated by RFC 1123. However, some systems do not handle blank envelope From: addresses properly and can require the use of this option.

Bit 1 (value = 2) controls whether or not PMDF replaces all blank envelope addresses with the address of the local postmaster. Again, this is used to accomodate incompliant systems that don't conform to RFC 821, RFC 822, or RFC 1123.

Note also that the `returnenvelope` channel keyword can be used to impose this sort of control on a per-channel basis.

RETURN_PERSONAL (string)

The RETURN_PERSONAL option specifies the personal name to use when PMDF generates postmaster messages, *e.g.*, bounce messages. By default, PMDF uses the string "PMDF e-Mail Interconnect".

RETURN_UNITS (0 or 1)

The time units used by the message return system is controlled with this option; that is, this option controls the interpretation of the values specified for the `notices` keyword. A value of 0 selects units of days; a value of 1 selects units of hours. By default, units of days are used.

On OpenVMS, if hours are selected, 1, then, after the next time the message return job is submitted, the message return job will begin running every hour.

On UNIX systems, the scheduling of the execution of the message return job is performed by changing the `crontab` entry controlling when it runs; see the appropriate edition of the *PMDF Installation Guide* for information on that `crontab` entry.

If you choose to set RETURN_UNITS=1, see also the discussion in Section 1.4.4.1.

USE_ERRORS_TO (0 or 1)

The USE_ERRORS_TO option controls whether or not PMDF makes use of the information contained in Errors-to: header lines when returning messages. Setting this option to 1 directs PMDF to make use of this header line. A value of 0, the default, disables use of this header line. Note that this default represents a change from the default in previous versions of PMDF.

USE_WARNINGS_TO (0 or 1)

The USE_WARNINGS_TO option controls whether or not PMDF makes use of the information contained in Warnings-to: header lines when returning messages. Setting this option to 1 directs PMDF to make use of these header lines. The default is 0, which disables use of this header line. Note that this default represents a changes from the default in previous versions of PMDF.

7.3.5 Message Size Options

This section lists options relating to message size, such as limits on the size of messages allowed in PMDF, message size affecting message processing priority, limits on the extent to which PMDF looks into messages of complex MIME structure, and fine tuning of message fragmentation.

BLOCK_LIMIT (integer > 0)
This option places an absolute limit on the size, in blocks, of any message which can be sent or received with PMDF. Any message exceeding this size will be rejected. By default, PMDF imposes no size limits. Note also that the `blocklimit` channel keyword can be used to impose limits on a per-channel basis. The size in bytes of a block is specified with the BLOCK_SIZE option.

BLOCK_SIZE (integer > 0)
PMDF uses the concept of a "block" in several ways. For example, the PMDF log files (resulting from placing the `logging` keyword on channels) record message sizes in terms of blocks. Message size limits specified via the `maxblocks` keyword are also in terms of blocks. Normally a PMDF block is equivalent to 1024 characters. This option can be used to modify this sense of what a block is. A good alternative might be 512, to match the OpenVMS definition of a block.

Note: PMDF stores message sizes internally as an integer number of blocks. If the size of a block in bytes is set to a very small value it is possible for a very large message to cause an integer overflow. A message size of greater than 2**31 blocks would be needed, but this value is not inconceivable if the block size is small enough.

BOUNCE_BLOCK_LIMIT (integer)
This option can be used to force bounces of messages over the specified size to return only the message headers, rather than the full message content.

CONTENT_RETURN_BLOCK_LIMIT (integer)
This option can be used to force on the NOTARY non-return of content flag for messages over the specified size; if such a message is subsequently bounced by a system that supports NOTARY, then the original message contents will not be included in the bounce message.

LINE_LIMIT (integer)
This option places an absolute limit on the overall number of lines in any message which can be sent or received with PMDF. Any message exceeding this limit will be rejected. By default, PMDF imposes no line count limits. Note also that the `linelimit` channel keyword can be used to impose limits on a per-channel basis.

MAX_HEADER_BLOCK_USE (real number between 0 and 1)
The MAX_HEADER_BLOCK_USE keyword controls what fraction of the available message blocks can be used by message headers. See Section 2.3.4.77 for additional information on how this option interacts with the `maxblocks` channel keyword.

MAX_HEADER_LINE_USE (real number between 0 and 1)
The MAX_HEADER_LINE_USE keyword controls what fraction of the available message lines can be used by message headers. See Section 2.3.4.77 for additional information on how this option interacts with the `maxlines` channel keyword.

MAX_INTERNAL_BLOCKS (integer)
The MAX_INTERNAL_BLOCKS option specifies how large (in PMDF blocks) a message PMDF will keep entirely in memory; messages larger than this size will be written to temporary files. The default is 30. For systems with lots of memory, increasing this value can provide a performance improvement.

MAX_MIME_LEVELS (integer)
Specify the maximum depth to which PMDF should process MIME messages. The default is 100, meaning that PMDF will process up to one hundred levels of message nesting. Higher values can require additional amounts of memory and, for the Dispatcher, additional per-thread storage space; see the discussion of the STACKSIZE Dispatcher option in Section 11.3.1.

MAX_MIME_PARTS (integer)
Specify the maximum number of MIME parts which PMDF should process in a MIME message. The default value is 0, meaning no limit is imposed.

NORMAL_BLOCK_LIMIT (integer)
The NORMAL_BLOCK_LIMIT option can be used to instruct PMDF to downgrade the priority of messages based on size: messages above the specified size will be downgraded to non-urgent priority. This priority, in turn, can affect whether the message is processed immediately, or whether it is left to wait for processing until the next periodic job runs; see Section 2.3.4.9. The value is interpreted in terms of PMDF blocks, as specified by the `BLOCK_SIZE` option. Note also that the `normalblocklimit` channel keyword can be used to impose such downgrade thresholds on a per-channel basis. Section 2.3.4.10,

NON_URGENT_BLOCK_LIMIT (integer)
The NON_URGENT_BLOCK_LIMIT option can be used to instruct PMDF to downgrade the priority of messages based on size: messages above the specified size will be downgraded to lower than non-urgent priority, meaning that they will not be processed immediately and will wait for processing until the next periodic job runs; see Section 2.3.4.9. The value is interpreted in terms of PMDF blocks, as specified by the `BLOCK_SIZE` option. Note also that the `nonurgentblocklimit` channel keyword can be used to impose such downgrade thresholds on a per-channel basis.

URGENT_BLOCK_LIMIT (integer)
The URGENT_BLOCK_LIMIT option can be used to instruct PMDF to downgrade the priority of messages based on size: messages above the specified size will be downgraded to normal priority. This priority, in turn, can affect whether the message is processed immediately, or whether it is left to wait for processing until the next periodic job runs; see Section 2.3.4.9. The value is interpreted in terms of PMDF blocks, as specified by the `BLOCK_SIZE` option. Note also that the `urgentblocklimit`, channel keyword can be used to impose such downgrade thresholds on a per-channel basis.

7.3.6 Logging, Monitoring, and Counters Options

The options listed in this section affect PMDF logging, monitoring, and counters. The CIRCUITCHECK_COMPLETED_BINS option relates to PMDF circuit check counter binning. The LOG_DELAY_BINS and LOG_SIZE_BINS options relate to PMDF counters binning. The LOG_SNDOPR option can be set by sites that want to have OPCOM messages (OpenVMS) or syslog messages (UNIX) or event log entries (NT) in cases of logging or counters updating problems. OpenVMS sites that have heavy message traffic

logged to the PMDF log file can find it useful to adjust the LOG_ALQ and LOG_DEQ option so that the underlying file allocation uses larger extents. The rest of these logging options affect the formatting of the PMDF log file and logging of optional additional information.

CIRCUITCHECK_COMPLETED_BINS (comma-separated list of up to eight integers)
This option specifies the bin divisions, in seconds, for the PMDF circuit check counters. The default values are 120, 300, 900, 1800, 3600, 7200, 14400, and 28800; *i.e.*, two minutes, five minutes, fifteen minutes, thirty minutes, one hour, two hours, four hours, and eight hours, respectively.

LOG_ALQ (integer; OpenVMS only)
The LOG_ALQ option specifies the default allocation quantity (in OpenVMS blocks) for the PMDF log file, `mail.log_current`. The default value is 2000, or twice the LOG_DEQ value if LOG_DEQ has been explicitly set. On a busy system that is updating that log file frequently, increasing this value can provide increased efficiency.

LOG_CONNECTION (integer)
The LOG_CONNECTION option controls whether or not connection information, *e.g.*, the domain name of the SMTP client sending the message, is saved in the `mail.log` file. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Bit	Value	Usage
0	1	When set, connection information is included in E, D and R log records.
1	2	When set, connection open/close/fail records are logged by message enqueue and dequeue agents such as the SMTP and X.400 clients and servers. This bit also enables use of the $T flag (for causing logging) in PORT_ACCESS rejection entries.
2	4	When set, `I` records are logged recording ETRN events.
4	16	When set, C entries can include site-supplied text from a PORT_ACCESS mapping table entry.

Bit 0 is the least significant bit.

Thus for instance enabling LOG_CONNECTION=3 will result both in additional sorts of log file entries—entries showing when an SMTP connection is opened or closed—and also additional information in regular log file entries showing the name of the system connecting (or being connected to), or the channel hostname of the enqueuing channel when the enqueuing channel is not an SMTP channel. (This is a change from PMDF V5.1 and earlier, where the value was simply 0 or 1, with 1 enabling all the then-available connection logging.) TCP/IP channels have a channel level option that can override this setting for particular channels; see Section 21.1.2.2. For examples of the sort of information resulting from setting LOG_CONNECTION=3, see Figure 31–10 and Figure 31–11.

LOG_CONNECTIONS_SYSLOG (integer; UNIX and NT only)
Send PMDF connection log file entries to syslog (UNIX) or event log (NT).

LOG_DELAY_BINS (comma-separated list of up to five integers)
This option specifies the bin divisions for the PMDF counters tracking numbers of messages delivered in the specified number of seconds. The defaults values are 60, 600, 6000, 60000, 600000.

LOG_DEQ (integer; OpenVMS only)

The LOG_DEQ option specifies the default extend quantity (in OpenVMS blocks) for the PMDF log file, `mail.log_current`. The default value is 1000. On a busy system that is updating that log file frequently, increasing this value can provide increased efficiency.

LOG_FILENAME (0 or 1)

The LOG_FILENAME option controls whether or not the names of the files in which messages are stored are saved in the `mail.log` file. A value of 1 enables file name logging. When file name logging is enabled, the file name will appear as the first field after the final form envelope To: address. A value of 0 (the default) disables file name logging.

LOG_FORMAT (1, 2, or 3)

The LOG_FORMAT option controls formatting options for the `mail.log` file. A value of 1 (the default) is the standard format. A value of 2 requests non-null formatting: empty address fields are converted to the string "<>". A value of 3 requests counted formatting: all variable length fields are preceded by "N:", where "N" is a count of the number of characters in the field.

LOG_HEADER (0 or 1)

The LOG_HEADER option controls whether PMDF writes message headers to the `mail.log` file. A value of 1 enables message header logging. The specific headers written to the log file are controlled by a site-supplied `log_header.opt` file. The format of this file is that of other PMDF header option files; see Section 2.3.7. For instance, a `log_header.opt` file containing

```
To: MAXIMUM=1
From: MAXIMUM=1
Defaults: MAXIMUM=-1
```

would result in writing the first To: and the first From: header per message to the log file.

A value of 0 (the default) disables message header logging.

LOG_LOCAL (0 or 1)

The LOG_LOCAL option controls whether or not the domain name for the local host is appended to logged addresses that don't already contain a domain name. A value of 1 enables this feature, which is useful when logs from multiple systems running PMDF are concatenated and processed. A value of 0, the default, disables this feature.

LOG_MESSAGE_ID (0 or 1)

The LOG_MESSAGE_ID option controls whether or not message IDs are saved in the `mail.log` file. A value of 1 enables message ID logging. When message ID logging is enabled, the message ID will be logged after the final form envelope To: address entry—and after the message file name, if LOG_FILENAME=1 is also enabled. A value of 0 (the default) disables message ID logging.

LOG_MESSAGES_SYSLOG (integer; UNIX and NT only)

Send PMDF message log file entries to syslog (UNIX) or event log (NT). A value of 1 adds entries to both the syslog/event log and to the mail.log file. A value of 2 adds entries only to syslog/event log (not to mail.log).

LOG_NODE (0 or 1; OpenVMS only)

The LOG_NODE option controls whether or not the node associated with a process that enqueues mail is saved in the `mail.log` file. This can be useful information when PMDF is running in a multi-node cluster. A value of 1 enables node name logging. When the node name is logged, it will appear as the first field following the date and time stamps in log entries. A value of 0 (the default) disables node name logging.

LOG_NOTARY (0 or 1)

The LOG_NOTARY option controls whether PMDF includes an indicator of NOTARY (delivery receipt) flags in the `mail.log` file entries. A value of 1 enables NOTARY flag logging. A value of 0 (the default) disables it. The NOTARY flags will be logged as a bit encoded integer after the current form of the envelope To: address.

LOG_PROCESS (0 or 1)

The LOG_PROCESS option controls whether or not the id of the process that enqueues mail is saved in the `mail.log` file. A value of 1 enables process id logging. A value of 0 (the default) disables it. The process id will be logged after the date and time stamps in log entries—and after the node name, if LOG_NODE=1 is also enabled. The process id field itself will consist of the process id in a hexadecimal representation followed by a period, next in the case of a multithreaded channel the thread id followed by a period, followed by a counter. That is, in the case of a single threaded channel

process-id.counter

or in the case of a multithreaded channel

process-id.thread-id.counter

Note in particular that via the process id and thread id, TCP/IP channel message enqueue/dequeue (E/D) records can be correlated with SMTP connection open/close (O/C) records.

LOG_SENSITIVITY (0 or 1)

The LOG_SENSITIVITY option controls whether message Sensitivity: header values are included in log entries. A value of 1 enables such logging; the default value of 0 disables such logging. If logging is enabled, the sensitivity value will be logged in an integer representation after the connection information, before the transport information.

LOG_SIZE_BINS (comma-separated list of up to five integers)

This option specifies the bin divisions for the PMDF counters tracking numbers of messages of the specified number of (PMDF) blocks. The default values are 2, 10, 50, 100, 500.

LOG_SNDOPR (0 or 1)

The LOG_SNDOPR option controls the production of OPCOM messages (OpenVMS) or syslog messages (UNIX) or event log entries (NT) by the PMDF message logging facility. If this feature is enabled by specifying a value of 1, the logging facility will produce a message if it encounters any difficulty writing to the log file. A value of 0 (the default) turns off these messages.

LOG_USERNAME (0 or 1)

The LOG_USERNAME option controls whether or not the username associated with a process that enqueues mail is saved in the `mail.log` file. Note that messages submitted via SMTP with authentication (SMTP AUTH) will be considered to be owned by the username that authenticated, prefixed with the asterisk, `*`, character. A value of 1 enables username logging. When username logging is enabled, the username will be

logged after the final form envelope To: address field in log entries—and after the message ID, if LOG_MESSAGE_ID=1 is also enabled. A value of 0 (the default) disables username logging.

SEPARATE_CONNECTION_LOG (0 or 1)

The SEPARATE_CONNECTION_LOG option controls whether the connection log information generated by setting LOG_CONNECTION=1 is stored in the usual PMDF message logging files, `mail.log*`, or stored separately in `connection.log*` files. SEPARATE_CONNECTION_LOG=0, the default, causes connection logging to be stored in the regular message log files; a value of 1 causes the connection logging to be stored separately.

SNDOPR_PRIORITY (integer)

Set the priority of operator broadcast or the syslog level of syslog messages or the severity of the NT event log entry.

7.3.7 Message Loop Detection and HELD Messages

This section lists options relating to PMDF's facility to sideline as `.HELD` messages that appear to be looping. See also Section 33.4.7 and Section 34.4.7 for discussions of such sidelined messages.

HELD_SNDOPR (0 or 1)

The HELD_SNDOPR option controls the production of OPCOM messages (OpenVMS) or syslog messages (UNIX) or event log entries (NT) when a message is forced into a held state because it has too many Received: header lines. (See the documentation on MAX_*RECEIVED_LINES options below for additional information.) A value of 1 instructs PMDF to issue a message when this happens. A value of 0 (the default) turns off these messages.

MAX_LOCAL_RECEIVED_LINES (integer)

As PMDF processes a message, it scans any Received: header lines attached to the message looking for references to the official local host name. (Any Received: line that PMDF inserts will contain this name). If the number of Received: lines containing this name exceeds the MAX_LOCAL_RECEIVED_LINES value, the message is entered into the PMDF queue in a held state. The default for this value is 10 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. See Section 33.4.7 or Section 34.4.7 for advice on dealing with held messages.

MAX_MR_RECEIVED_LINES (integer)

As PMDF processes a message, it counts the number of MR-Received: header lines in the message's header. If the number of MR-Received: lines exceeds the MAX_MR_RECEIVED_LINES value, the message is entered into the PMDF queue in a held state. The default for this value is 20 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. See Section 33.4.7 or Section 34.4.7 for advice on dealing with held messages.

MAX_RECEIVED_LINES (integer)
As PMDF processes a message, it counts the number of Received: header lines in the message's header. If the number of Received: lines exceeds the MAX_RECEIVED_LINES value, the message is entered into the PMDF queue in a held state. The default for this value is 50 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. See Section 33.4.7 or Section 34.4.7 for advice on dealing with held messages.

MAX_TOTAL_RECEIVED_LINES (integer)
As PMDF processes a message, it counts the number of Received:, MR-Received:, X400-Received: header lines in the message's header. If the number of all such header lines exceeds the MAX_TOTAL_RECEIVED_LINES value, the message is entered into the PMDF queue in a held state. The default for this value is 100 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. See Section 33.4.7 or Section 34.4.7 for advice on dealing with held messages.

MAX_X400_RECEIVED_LINES (integer)
As PMDF processes a message, it counts the number of X400-Received: header lines in the message's header. If the number of Received: lines exceeds the MAX_X400_RECEIVED_LINES value, the message is entered into the PMDF queue in a held state. The default for this value is 50 if no value is specified in the option file. This check blocks certain kinds of message forwarding loops. The message must be manually moved from the held state for processing to continue. See Section 33.4.7 or Section 34.4.7 for advice on dealing with held messages.

7.3.8 File Format and File Handling Options

The options described in this section affect the format of various PMDF files and affect the handling of message files.

COMMENT_CHARS (integer list {33, 59})
This option controls what characters are taken to signal a comment when they appear in the first column of various PMDF input files. The value of this option takes the form of a list of ASCII character values in decimal. The default is the list {33, 59}, which specifies exclamation points and semicolons as comment introduction characters.

FSYNC (0 or 1; UNIX and NT only)
On UNIX and NT platforms, the FSYNC option can be used to cause PMDF to use the fsync function (UNIX) or FlushFileBuffers function (NT) to flush disk output when closing a message file. If such flushing is not performed explicitly by PMDF, it is left up to the O/S to perform on its own timetable; potentially, if a UNIX or NT system crashes at just the wrong moment, messages not yet synched to disk could be lost. The tradeoff, however, is that performing explicit flushing for every message incurs a performance penalty. FSYNC=1, meaning that flushes are performed explicitly by PMDF, is the default, ensuring message safety at the expense of a performance hit.

7.3.9 Internal Size Options

This section describes PMDF options relating to internal PMDF sizing issues. In general these options should not be set manually, but should instead be automatically resized when necessary by using the `pmdf cnbuild` utility, as described in Section 8.1.4.

ALIAS_HASH_SIZE (integer <= 32,767)
This option sets the size of the alias hash table. This in turn is an upper limit on the number of aliases that can be defined in the alias file. The default is 256; the maximum value allowed is 32,767.

ALIAS_MEMBER_SIZE (integer <= 30,000)
This option controls the size of the index table that contains the list of alias translation value pointers. The total number of addresses on the right hand sides of all the alias definitions in the alias file cannot exceed this value. The default is 320; the maximum allowed is 30,000.

CHANNEL_TABLE_SIZE (integer <= 32,767)
This option controls the size of the channel table. The total number of channels in the configuration file cannot exceed this value. The default is 256; the maximum is 32,767.

CIRCUITCHECK_PATHS_SIZE (integer <= 256)
This option controls the size of the circuit check paths table, and thus the total number of circuit check configuration file entries. The default is 10.

CONVERSION_SIZE (integer <= 2000)
This option controls the size of the conversion entry table, and thus the total number of conversion file entries cannot exceed this number. The default is 32.

DOMAIN_HASH_SIZE (integer <= 32,767)
This option controls the size of the domain rewrite rules hash table. Each rewrite rule in the configuration file consumes one slot in this hash table, thus the number of rewrite rules cannot exceed this option's value. The default is 512; the maximum number of rewrite rules allowed is 32,767.

HOST_HASH_SIZE (integer <= 32,767)
This option controls the size of the channel hosts hash table. Each channel host specified on a channel definition in the PMDF configuration file (both official hosts and aliases) consumes one slot in this hash table, so the total number of channel hosts cannot exceed the value specified. The default is 512; the maximum value allowed is 32,767.

MAP_NAMES_SIZE (integer > 0)
The MAP_NAMES_SIZE option specifies the size of the mapping table name table, and thus the total number of mapping tables cannot exceed this number. The default is 32.

STRING_POOL_SIZE (integer <= 10,000,000)
The STRING_POOL_SIZE option controls the number of character slots allocated to the string pool used to hold rewrite rule templates, alias list members, mapping entries, *etc.* A fatal error will occur if the total number of characters consumed by these parts of the configuration files exceeds this limit. The default is 65,000; the maximum allowed value is 10,000,000.

WILD_POOL_SIZE (integer <= 200,000)

The WILD_POOL_SIZE option controls the total number of patterns that can appear throughout mapping tables. A fatal error will occur if the total number of mapping patterns exceeds this limit. The default is 8,000; the maximum allowed value is 200,000.

7.3.10 Debugging Options

This section lists options for enabling debugging of various PMDF facilities.

DEQUEUE_DEBUG (0 or 1)

This option specifies whether or not debugging output from PMDF's dequeue facility QU is produced. If enabled with a value of 1, this output will be produced on all channels that use the QU routines. The default value of 0 disables this output.

POST_DEBUG (0 or 1)

This option specifies whether or not debugging output is produced by PMDF's periodic delivery job. If enabled with a value of 1, this output will be produced in the `post.log` file. The default value of 0 disables this output.

On UNIX, when debugging the periodic delivery job it can also be useful to set the PMDF tailor option PMDF_POST_VERIFY=1, to cause the delivery job script to echo which steps it has performed to the log file.

RETURN_DEBUG (0 or 1)

The RETURN_DEBUG option enables or disables debugging output in the nightly message bouncer batch job. A value of 0 disables this output (the default) while a value of 1 enables it. Debugging output, if enabled, appears in the output log file, if such a log file is present. However, on OpenVMS, the output log file is customarily discarded once the batch job terminates; thus to retain the output log file for inspection, the usual approach is to set RETURN_DEBUG and then use the SET ENTRY DCL command to modify the batch job itself to preserve the log file. On UNIX, the presence of an output log file is controlled by the `crontab` entry for the return job; see the appropriate edition of the *PMDF Installation Guide* for more details. On NT, the presence of an output log file is controlled by the Scheduler entry that schedules the running of the PMDF return job.

On UNIX and NT, when debugging the periodic return job it can also be useful to set the PMDF tailor option (UNIX) or Registry entry (NT) PMDF_RETURN_VERIFY=1, to cause the delivery job script to echo which steps it has performed to the log file.

7.3.11 Options for OpenVMS User Agents

This section lists options relating to OpenVMS user agents such as VMS MAIL, PMDF MAIL, and DECwindows MAIL.

DELIVERY_RECEIPT_OFF (string; OpenVMS only)

This option is used to specify a special RFC 822 comment string used in IN% addresses in VMS MAIL to disable any requests for a delivery receipt. The default if this option is not specified is:

```
(NO-DELIVERY-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

DELIVERY_RECEIPT_ON (string; OpenVMS only)

This option is used to specify a special RFC 822 comment string used in IN% addresses in VMS MAIL to request a delivery receipt. The default if this option is not specified is:

```
(DELIVERY-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

DIS_NESTING (non-negative integer; OpenVMS only)

The DIS_NESTING option controls how many nesting levels PMDF allows in the expansion of VMS MAIL distribution lists. A value of 0 disables the ability to use VMS MAIL distribution lists. Currently this option only affects the PMDF MAIL utility. The default value for this option is 20.

FORM_NAMES (list of comma separated strings; OpenVMS only)

Multiple values should be separated with commas but *not* with spaces. The default is "FAX-FORM,PH-FORM,X500-FORM".

MAIL_DELIVERY_FILENAME (string)

The MAIL_DELIVERY_FILENAME option sets the actual filename used by users to store message delivery option information. The default name is `mail.delivery` if this option is not used to change the name. No device or directory can be specified here; these files are always located in users' home directories.

MISSING_ADDRESS (string)

This option can be used to specify the address to insert in the VMS MAIL From: header if no From: address was originally present; the default is missing-address@no.where.

MP_SIGNED_MODE (0 or 1)

The MP_SIGNED_MODE option controls whether or not PMDF interprets (parses) messages of type multipart/signed. The default is 0, which means PMDF treats the message as opaque. The risk in parsing multipart/signed messages is that the exact content would most likely change when PMDF re-assembles the message, thus invalidating the signature.

MULTINET_MM_EXCLUSIVE (-3 to 3; OpenVMS only)

This option is documented for historical reasons only; delivery to MultiNet MM is no longer supported. (Prior to PMDF V5.2, the local channel supported delivery to MultiNet MM as an alternative to delivery to VMS MAIL.)

The MULTINET_MM_EXCLUSIVE option controls whether or not PMDF's local channel delivers exclusively to MultiNet MM or not. Possible values are:

Value	Usage
-3	Never deliver mail to MultiNet MM mailboxes.
-2	Not used at present; do not specify.
-1	Deliver mail to MultiNet MM mailboxes only if MultiNet MM is specified as part of the user's profile information.
0	Deliver mail to MultiNet MM mailboxes if the user has the appropriate MultiNet MM mailbox files in his or her home directory.
+1	Deliver mail to the user's Multinet MM mailbox by default unless the user's profile information specifies some other type of delivery.
+2	Not used at present; do not specify.
+3	Deliver mail to the user's Multinet MM mailbox unconditionally.

The default is -1, which means that MultiNet MM delivery will be used only when the user has profiled MultiNet MM as his or her preferred mailbox format.

READ_RECEIPT_OFF (string; OpenVMS only)

This option is used to specify a special RFC 822 comment string used in IN% addresses in VMS MAIL to disable any requests for a read receipt. The default if this option is not specified is:

```
(NO-READ-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

READ_RECEIPT_ON (string; OpenVMS only)

This option is used to specify a special RFC 822 comment string used in IN% addresses in VMS MAIL to request a read receipt. The default if this option is not specified is:

```
(READ-RECEIPT)
```

The enclosing parentheses are required and must be specified in the option value.

SAFE_TCL_MODE (0–3)

This option is a bit encoded integer. The lowest bit, when set, allows components of PMDF to interpret message parts of type application/safe-tcl upon user confirmation. (At present, PMDF MAIL is the only component of PMDF which supports Safe-Tcl.) The second lowest bit, when set, puts PMDF's Safe-Tcl interpreter into a very paranoid mode in which it will not allow information from sensitive message header lines to be disclosed to Safe-Tcl scripts.

The default value for this option is 3 which allows Safe-Tcl scripts to be executed upon user confirmation, but does so in "paranoid" mode.

USE_MAIL_DELIVERY (0 or 1)

The USE_MAIL_DELIVERY option controls whether or not PMDF checks to see if a given user has a `mail.delivery` file in their home directory and, if the file exists, uses it to direct message delivery operations on the local channel for the user. The default is 1, which means that PMDF will check for these files. A value of 0 will disable consultation of `mail.delivery` files.

VMS_MAIL_EXCLUSIVE (-3 to 3; OpenVMS only)

This option is documented for historical reasons only; delivery to MultiNet MM is no longer supported. (Prior to PMDF V5.2, the local channel supported delivery to MultiNet MM as an alternative to delivery to VMS MAIL.)

The VMS_MAIL_EXCLUSIVE option controls whether or not PMDF's local channel delivers exclusively to VMS MAIL or not. Possible values are:

Value	Usage
-3	Never deliver mail to VMS MAIL mailboxes.
-2	Not used at present; do not specify.
-1	Deliver mail to VMS MAIL mailboxes only if VMS MAIL is specified as part of the user's profile information.
0	Deliver mail to VMS MAIL mailboxes unless the user has some other sort of mailbox in his or her home directory or has specified some other kind of delivery as part of his or her profile information.
+1	Deliver mail to the user's VMS MAIL mailbox by default unless the user's profile information specifies some other type of delivery.
+2	Not used at present; do not specify.
+3	Deliver mail to the user's VMS MAIL mailbox unconditionally.

The default is 0, which means that either VMS MAIL is used unless a given user has the necessary MultiNet MM files in his or her home directory or has profiled the use of MultiNet MM.

7.3.12 Miscellaneous Options

This section contains other options.

INCLUDE_CONNECTIONINFO (integer)

The INCLUDE_CONNECTIONINFO option provides a means of including the transport and application connection information in various mapping probes that otherwise would not include this material. If included, the information appears at the beginning of the mapping probe in the same format used in the `FROM_ACCESS`, `MAIL_ACCESS`, and `ORIG_MAIL_ACCESS` mappings. The value of the option is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below. The default is 0. Each currently defined bit corresponds to a particular non-positional alias parameter.

Bit	Value	Mapping
0	1	AUTH_MAPPING
1	2	MODERATOR_MAPPING
2	4	CANT_MAPPING
5	32	HOLD_MAPPING
6	64	NOHOLD_MAPPING

Bit 0 is the least significant bit.

ACCESS_ORCPT (0 or 1)

The ACCESS_ORCPT option provides a means of including the original recipient address (the value of the ORCPT option in the SMTP protocol exchange) in the probes for the SEND_ACCESS and ORIG_SEND_ACCESS mapping tables. The default is 0, which means that the ORCPT value is not included in the probe. If this option is set to 1, the ORCPT value appears at the end of the mapping table probe.

8 Maintaining the Configuration

The critical PMDF configuration files,

- the configuration file, discussed in Section 1.2, Section 2.2 and Section 2.3,
- the alias file, discussed in Chapter 3,
- the mappings file, discussed in Chapter 5,
- the conversion file, discussed in Chapter 6,
- the option file, discussed in Chapter 7,
- the security configuration file, discussed in Chapter 14,
- the system wide filter file, discussed in Section 16.2.3, and
- the circuit check configuration file, discussed in Section 31.3.1.1,

may be pre-processed and compiled into a single image referred to as a "compiled configuration". By pre-processing these files and storing them into an image, the initialization time for PMDF is significantly reduced thereby improving PMDF's performance. Compiling the configuration will be discussed in Section 8.1.

When updating the PMDF configuration, regardless of whether the PMDF configuration is compiled, it is important to ensure that any affected components of PMDF are made aware of the change. That is, if the PMDF configuration is not compiled, then components need to be restarted after changes to the PMDF configuration files; if a compiled configuration is used, then the configuration needs to be recompiled after changes and *then* components need to be restarted. An overview of what components need to be restarted when various changes are made to the PMDF configuration is provided in Section 8.2.

8.1 Compiling the Configuration

PMDF contains a utility, `cnbuild`, to compile the configuration, option, mapping, conversion, alias, security, and system wide filter configuration files into a single shareable image (on OpenVMS) or a single image in shared memory (on UNIX) or a dynamic link library (on Windows). The main reason for compiling configuration information is simple: performance. [1] Another feature of using a compiled configuration is that you can test configuration changes more conveniently, since the configuration files themselves are not "live" when a compiled configuration is in use.

[1] A test on a µVAX II system showed that reading a fairly complex configuration and alias file took about 6 seconds of CPU time. By contrast, initialization of PMDF took about 1 second of CPU time when the data was precompiled as a shareable image.

Whenever a component of PMDF (*e.g.,* a channel program) must read the configuration file it first checks to see if a compiled configuration exists. If it does, the image is merged into the running program (on OpenVMS) or attached to by the running program (on UNIX or Windows). [2] If the attempt to load the compiled configuration fails for any reason, PMDF falls back on the old method of reading the text files instead.

Note: The only penalty paid for compilation is the need to rebuild and, on OpenVMS systems, reinstall the image every time the configuration, option, mapping, conversion, alias, or security configuration files are edited.

On OpenVMS systems, the image file is referenced with the PMDF_CONFIG_DATA logical. This logical translates to `PMDF_EXE:CONFIG_DATA.EXE`; PMDF_EXE itself is a logical which translates to `PMDF_ROOT:[xxx_EXE]` depending upon the architecture (VAX, Alpha, or IA64). The image file can be installed with the standard OpenVMS INSTALL utility.

On UNIX systems, the name of the image file is specified with the PMDF_CONFIG_DATA option in the PMDF tailor file. By default, this is the file `/pmdf/lib/config_data`.

On Windows systems, the name of the image file is specified with the PMDF_CONFIG_DATA PMDF Tailor Registry entry. Usually this is the file `C:\pmdf\lib\config_data`.

Note: As always when there is a change to the PMDF configuration, resident PMDF processes (such as the multithreaded SMTP server) should be restarted with the `pmdf restart` command.

8.1.1 Compiling the Configuration on OpenVMS

The `pmdf cnbuild` utility on OpenVMS is used to compile a PMDF configuration (configuration file, alias file, PMDF option file, mapping file, conversion file, security configuration file, and system wide filter file). After you are satisfied with your configuration file, alias file, option file, mapping file, conversion file, security configuration file, and system wide filter file, issue the OpenVMS command

```
$ PMDF CNBUILD
```

to compile your configuration into a loadable, shareable image. The alias, option, mapping, conversion, security configuration, system wide filter, and configuration files will be read and the compiled configuration image will be created. Any errors detected in the files will be reported and will cause the compilation to abort without producing an image.

When CNBUILD is run, a shareable image is created that is suitable for use on the architecture that you ran `cnbuild` on (VAX, Alpha, or I64). To generate shareable images for the other architectures, you must run `cnbuild` on systems of each of those other architectures.

[2] There are two exceptions to this rule. The first is the `pmdf cnbuild` utility itself, which for obvious reasons always reads the text files and never tries to load the image form of the configuration data. The other exception is the `pmdf test` utility, which may be instructed to ignore any compiled image with the /NOIMAGE_FILE (OpenVMS) or `-noimage_file` (UNIX or Windows) qualifier, which is useful for testing changes prior to compiling them.

Once the image has been produced it must be installed for proper operation:

```
$ INSTALL CREATE PMDF_CONFIG_DATA/OPEN/SHARED/HEADER
```

If the image already exists and is being updated, use the command:

```
$ INSTALL REPLACE PMDF_CONFIG_DATA
```

In a cluster environment, the image should be replaced on every system that is running PMDF. You may also want to purge old copies of the image in `PMDF_EXE:`, since the files are fairly large.

At system startup, the PMDF startup procedure will install the PMDF_CONFIG_DATA image if one exists. There is no need to add the INSTALL command shown above to your system startup procedure.

Refer to Chapter 29 for complete details on the use of the CNBUILD utility.

Note: As always when there is a change to the PMDF configuration, *e.g.*, a new compiled configuration, any resident PMDF processes such as the multithreaded SMTP server, POP3 server, IMAP server, BN_SLAVE, or FAX_RECEIVE should be restarted with the PMDF RESTART command.

8.1.2 Compiling the Configuration on UNIX

The `pmdf cnbuild` utility on UNIX is used to compile a PMDF configuration (configuration file, alias file, PMDF option file, mapping file, conversion file, security configuration file, and system wide filter file). After you are satisfied with your configuration file, alias file, option file, mapping file, conversion file, security configuration file, and system wide filter file, issue the command

```
# pmdf cnbuild
```

to compile your configuration into a single image and load the resulting image into shared memory. The alias, option, mapping, conversion, security configuration, system wide filter, and configuration files will be read and the compiled configuration image will be created. Any errors detected in the files will be reported and will cause the compilation to abort without producing an image.

Refer to Chapter 30 for complete details on the use of the `cnbuild` utility.

Note: As always when there is a change to the PMDF configuration, such as a new compiled configuration, any resident PMDF processes such as POP, IMAP, and TCP SMTP servers should be restarted with the `pmdf restart` command.

8.1.3 Compiling the Configuration on Windows

The `pmdf cnbuild` utility on Windows is used to compile a PMDF configuration (configuration file, alias file, PMDF option file, mapping file, conversion file, security configuration file, and system wide filter file). After you are satisfied with your configuration file, alias file, option file, mapping file, conversion file, security configuration file, and system wide filter file, issue the command

```
C:\> pmdf cnbuild
```

to compile your configuration into a single dynamic link library. The alias, option, mapping, conversion, security configuration, system wide filter, and configuration files will be read and the compiled configuration image will be created. Any errors detected in the files will be reported and will cause the compilation to abort without producing an image.

Refer to Chapter 30 for complete details on the use of the `cnbuild` utility.

Note: As always when there is a change to the PMDF configuration, such as a new compiled configuration, any resident PMDF processes such as the Dispatcher and Job Controller should be restarted with the `pmdf restart` command.

8.1.4 Extending Table Sizes

PMDF uses a variety of internal tables to store configuration information. The sizes of these tables are specified by various option file options. If no specifications are given, default sizes are used. The result for small configurations can be wasted memory, while for large configurations PMDF may fail with a table overflow error.

The solution to these problems is to specify larger or smaller tables using the option file. The option file controls the size of the various tables very precisely; the result will be optimum memory usage with no overflows.

The `cnbuild` utility can be used to generate such specifications in the option file automatically. This use of `cnbuild` does not mean that the configuration must be compiled. Either `cnbuild` is quite capable of generating such an option file without generating a corresponding compiled configuration. The details of this use of `cnbuild` are described below.

The size of the tables `cnbuild` creates in the precompiled image is set before `cnbuild` actually reads the configuration and alias files. As a result, small configurations generate images with lots of wasted space in them, while large configurations may exceed the default sizes of the tables and cause PMDF to report the infamous “no room in table” error message.

The solution to these problems is to generate a PMDF option file that describes the size of your configuration. The PMDF option file is described in Chapter 7. However, it is not necessary to know the format of that file; `cnbuild` is capable of building an option file for you.

To generate an option file that specifies proper table sizes to hold your configuration, use the OpenVMS command,

```
$ PMDF CNBUILD/NOIMAGE/MAXIMUM/OPTION
```

or the UNIX or Windows command,

```
pmdf cnbuild -noimage_file -maximum -option_file
```

Such a command will create a new PMDF option file with appropriate table sizes.

If you use a compiled configuration, you must then recompile your configuration. Consult Section 8.1 for details on compiling your configuration.

If you continue to get "no room in table" sorts of errors after using `cnbuild` to resize your PMDF option file (and after then recompiling, if you use a compiled configuration), then you likely have a configuration syntax error that is causing PMDF to believe it sees spurious configuration entries; see some of the suggestions in Section 33.3.1 or Section 34.3.1 for possible syntax errors for which to check.

Note that you only have to resize your configuration when its size changes enough to warrant it. It is not necessary to do this when minor changes are made; the size information output by `cnbuild` leaves room for moderate changes without resizing. The OpenVMS command,

```
$ PMDF CNBUILD/NOIMAGE/STATISTICS
```

or the UNIX or Windows command

```
pmdf cnbuild -noimage_file -statistics
```

may be used to determine exactly how close your configuration really is to the current sizes set in the option file.

8.2 Restarting After Configuration Changes

In general, if you are using a compiled PMDF configuration then you will need to recompile it after any changes to the files that comprise the PMDF compiled configuration: the PMDF configuration file and any files it includes, the alias file and any files it includes (but *not* alias database or mailing list files), the option file and any files it includes, the mappings file and any files it includes, the conversions file and any files it includes, the system wide filter file, the circuit check configuration file, and the security configuration file and any files it includes.

In general, you will need to restart any resident processes after any changes of which that resident process needs to be aware; such changes may include changes to the basic PMDF configuration (such as the changes above requiring recompilation of a compiled configuration) and may also include other, component-specific changes.

8.2.1 Restarting Specific Components

Table 8–1 provides an overview of when specific components should be restarted. For restarting PMDF components, use the PMDF RESTART (OpenVMS) or `pmdf restart` (UNIX or NT) utility, described in Chapter 29 or Chapter 30, respectively.

Table 8–1 Restarting Components

Component	Restart after changes to:
Circuit check	• The PMDF configuration[1], especially (but not only) the circuit check configuration file
Dispatcher	• The Dispatcher configuration file, `dispatcher.cnf`, and files it includes • The PORT_ACCESS mapping table in the PMDF mapping file • On OpenVMS, the PMDF_TIMEZONE logical name's value[6]
SMTP server	• The PMDF configuration[1] • The TCP/IP channel option file • New (replaced) PMDF databases[2]
HTTP server	• The HTTP server configuration file, `http.cnf` • The Dispatcher configuration file, `dispatcher.cnf`, and files it includes for changes affecting the HTTP service definition • The HTTP_ACCESS mapping table in the PMDF mapping file • Add the `filter`, `destinationfilter`, `sourcefilter`, or related channel keywords to a channel, or create a new system wide filter file

IMAP server	• The IMAP server configuration file, `imapd.cnf`
	• The Dispatcher configuration file, `dispatcher.cnf`, and files it includes for changes affecting the IMAP service definition
	• Initial creation of the PMDF password database
	• Conversion from PMDF V5.0 or earlier password database format to PMDF V5.1 and later format
	• On UNIX, initial creation of the PMDF profile database
	• The PMDF configuration[1] as it relates to the IMAP server, particularly:
	  • The security configuration file
	  • The PORT_ACCESS mappings authentication control features
	  • On OpenVMS, certain channel keyword changes on the local channel definition

[1]Here the PMDF configuration comprises the base PMDF configuration files or compiled configuration, *i.e.*, the PMDF configuration, alias, mappings, conversion, option, system wide filter, and security configuration files.

[2]While a new version of a PMDF database requires restarting PMDF components— for instance, a new version of the PMDF alias database requires restarting the SMTP server in order for the SMTP server to see the new version—a database updated "in place", via for instance the PMDF CRDB/APPEND (OpenVMS) or `pmdf crdb -append` (UNIX and NT) command does *not* require restarting any components, particularly not the SMTP server.

Table 8–1 (Cont.) Restarting Components

Component	Restart after changes to:

POP server	• The POP server configuration file, `pop3d.cnf`
	• The Dispatcher configuration file, `dispatcher.cnf`, and files it includes for changes affecting the POP3 service definition
	• Initial creation of the PMDF password database
	• Conversion from PMDF V5.0 or earlier password database format to PMDF V5.1 and later format
	• On UNIX, initial creation of the PMDF profile database
	• The PMDF configuration[1] as it relates to the IMAP server, particularly:
	  • The security configuration file
	  • The PORT_ACCESS mappings authentication control features
	  • On OpenVMS, certain channel keyword changes on the local channel definition

POPPASSD server	• The Dispatcher configuration file, `dispatcher.cnf`, and files it includes for changes affecting the POPPASSD service definition
	• Initial creation of the PMDF password database
	• Conversion from PMDF V5.0 or earlier password database format to PMDF V5.1 and later format
	• The PMDF configuration[1] as it relates to the POPPASSD server, particularly:
	  • The security configuration file
	  • The PORT_ACCESS mappings authentication control features

Lotus Notes channel servers	• The PMDF configuration[1] • The Dispatcher configuration file, `dispatcher.cnf`, and files it includes for changes affecting the LN service definitions • New (replaced) PMDF databases[2] • Lotus Notes channel option files
OpenVMS only	
Counters synchronization process	• The PMDF_COUNTER_INTERVAL logical

[1]Here the PMDF configuration comprises the base PMDF configuration files or compiled configuration, *i.e.*, the PMDF configuration, alias, mappings, conversion, option, system wide filter, and security configuration files.

[2]While a new version of a PMDF database requires restarting PMDF components— for instance, a new version of the PMDF alias database requires restarting the SMTP server in order for the SMTP server to see the new version—a database updated "in place", via for instance the PMDF CRDB/APPEND (OpenVMS) or `pmdf crdb -append` (UNIX and NT) command does *not* require restarting any components, particularly not the SMTP server.

Table 8–1 (Cont.) Restarting Components

Component	Restart after changes to:
	OpenVMS only
Process Symbiont[3]	• The Process Symbiont option file • New MR_* channels or changes in the `user` or `daemon` channel keyword clauses on MR_* channels
FAX_RECEIVE	• The PMDF configuration[1], at least any changes affecting PMDF-FAX • The FAX_TO_DATA channel option file
MailWorks server, ALL-IN-1 Sender, ALL-IN-1 Fetcher[4] [5]	• The PMDF configuration[1] • MRIF channel option files • New (replaced) PMDF databases[2] • The PMDF_TIMEZONE logical name's value[6]
Queue to e-mail symbiont[3]	• The Q2EMAIL option file
User agents, *e.g.*, PMDF MAIL, VMS MAIL, PMDF Pine	• The PMDF configuration[1] • New (replaced) PMDF databases[2] • The PMDF_TIMEZONE logical name's value[6]
	UNIX and NT
Job Controller	• The Job Controller configuration files, `job_controller.cnf` and `job_controller.cnf_site`

[1]Here the PMDF configuration comprises the base PMDF configuration files or compiled configuration, *i.e.*, the PMDF configuration, alias, mappings, conversion, option, system wide filter, and security configuration files.

[2]While a new version of a PMDF database requires restarting PMDF components— for instance, a new version of the PMDF alias database requires restarting the SMTP server in order for the SMTP server to see the new version—a database updated "in place", via for instance the PMDF CRDB/APPEND (OpenVMS) or `pmdf crdb -append` (UNIX and NT) command does *not* require restarting any components, particularly not the SMTP server.

[3]To restart symbionts, use DCL commands: STOP/QUEUE/RESET and START/QUEUE.

[4]Must be restarted using the server's own restart mechanism.

[5]If PMDF_TIMEZONE is not defined, PMDF looks first for SYS$LOCALTIME, then for SYS$TIMEZONE_DIFFERENTIAL and SYS$TIMEZONE_NAME.

9 The PMDF Process Symbiont (OpenVMS)

Note: The content of this chapter is only applicable to OpenVMS systems.

During operation, PMDF frequently handles messages by scheduling processes through OpenVMS batch queues. PMDF uses OpenVMS queues because they offer enormous advantages, including flexibility in scheduling, cluster-wide submission of processes, and cluster-wide load balancing through the use of generic and execution queues. Batch jobs, however, do present overhead that can be significant in heavy message load environments. That overhead generally comes from scheduler processing within the VMS print/batch subsystem and from process creation as PMDF batch jobs execute. Batch jobs also write log files, even when such log files are not necessary.

Overhead due to OpenVMS queue handling has been significantly reduced as of OpenVMS V5.5. Overhead due to process creation can be significantly reduced by using the PMDF Process Symbiont instead of standard batch queues for scheduling the execution of PMDF processes.

The PMDF Process Symbiont has been implemented as a multi-threaded server symbiont. A single Process Symbiont is capable of handling up to 32 execution queues. A job submitted to a Process Symbiont queue will be executed by a detached process created by the symbiont as needed. Unlike a standard batch job, however, the server process will not be deleted when the job completes. The server process will wait for the symbiont to present it with another command. If the symbiont does not reuse a server process within a timeout period, the server process will exit in order to free up a process slot. Process creation overhead is reduced the most during periods of heavy PMDF load as multiple messages can be handled by the server process without the timeout period expiring. Since Process Symbiont queues are also careful to generate channel job log files only when (a) an error occurs that needs to be logged, or (b) debugging is enabled for the channel, use of Process Symbiont queues also reduces the number of log files written.

9.1 Symbiont Configuration

The PMDF CONFIGURE QUEUES utility is a convenient way to configure PMDF to use PMDF Process Symbiont queues; see Section 9.1.1 below. Of course, you can also manually define queues that use the PMDF Process Symbiont; see Section 9.1.2 below.

Due to the limitations of OpenVMS server symbionts, a PMDF Process Symbiont queue cannot execute more than one job at a time. Therefore, for adequate message throughput, multiple simultaneous jobs should be allowed by creating several PMDF Process Symbiont execution queues with a single generic queue referencing all of them. Jobs submitted to the generic queue will be scheduled to run on the execution queues in parallel. Since PMDF submits jobs to MAIL$BATCH by default, that is typically used as the name of the generic queue.

Note that the benefits inherent in reusing server processes can be reduced if you create too many parallel execution queues. You need to balance your use of parallel and serial jobs in order to maximize throughput. You should see peak optimization when jobs can be scheduled across all parallel execution queues without pending too long in the generic queue. On the other hand, new jobs should still enter each execution queue within the server timeout period so that each execution queue's cached process is given something to do before timing out and deleting itself.

Symbiont server processes log output to the file

```
PMDF_LOG:task_server_queue-name.log
```

where *queue-name* is the name of a specific server queue. Command procedures executed by the server process continue to write their log output to files in the PMDF log directory such as `post.log` and `return.log`.

9.1.1 The PMDF Queue Configuration Utility

The PMDF CONFIGURE QUEUES utility configures a generic MAIL$BATCH queue to run a site-specified number of PMDF Process Symbiont queues. The PMDF CONFIGURE QUEUES utility will ask you how many Process Symbiont queues you want to run, and which node you want to run each on. The utility then generates the following files in `SYS$STARTUP:`, `pmdf_init_queues.com`, `pmdf_start_queues.com`, `pmdf_stop_queues.com`, and `pmdf_delete_queues.com`.

Example 9–1 shows a sample PMDF queue configuration utility dialogue.

Example 9–1 Example Process Symbiont Queue Configuration Dialogue on a Node ALPHA1

```
$ PMDF CONFIGURE QUEUES
This utility will create the following command files
        SYS$STARTUP:PMDF_INIT_QUEUES.COM
        SYS$STARTUP:PMDF_START_QUEUES.COM
        SYS$STARTUP:PMDF_STOP_QUEUES.COM
        SYS$STARTUP:PMDF_DELETE_QUEUES.COM

How many execution queues do you want? [4] 8
Do you want all queues to run on this node ALPHA1? [Y] Yes
Do you want to initalize the queues now? [Y] Yes
Do you want to start the queues now? [Y] Yes
```

The generated `pmdf_init_queues.com` procedure can be used to initialize your PMDF queues, and the `pmdf_start_queues.com` procedure can be used to start your PMDF queues. Such commands should be added to your system startup procedure. Generally PMDF queues are initialized early during system startup (before your networks are started) and these PMDF queues are then started a little later (after your networks are started).

pmdf_stop_queues.com and pmdf_delete_queues.com are provided for convenience in testing, but are not normally used in production; in particular, there is no need to stop PMDF queues before shutting down a system.

9.1.2 Manually Configuring PMDF Process Symbiont Queues

A PMDF Process Symbiont queue is created by initializing a server queue with pmdf_process_smb as its processor on the appropriate cluster node, *node*:[1]

```
$ INITIALIZE/QUEUE/DEVICE=SERVER/NOENABLE_GENERIC -
$_          /PROCESSOR=pmdf_process_smb/ON=node:: -
$_          /PROTECTION=(S:RWE,O:RWD,G:R,W:R) PMDF_1
```

Note that the /NOENABLE_GENERIC qualifier should be specified in order to prevent generic printer queues without specifically defined execution queues from unintentionally printing to this queue. Similarly, the queue should be protected from non-privileged submissions to prevent users from using the queue for jobs that are not PMDF jobs.

For example, the following sequence of commands will create four parallel Process Symbiont execution queues serving a single MAIL$BATCH generic queue. An example of these commands can be found in the file PMDF_COM:init_mail_queues.com-sample.

```
$ INITIALIZE/QUEUE/DEVICE=SERVER/NOENABLE_GENERIC -
$_          /PROCESSOR=pmdf_process_smb/ON=node:: -
$_          /PROTECTION=(S:RWE, O:RWD, G:R, W:R) PMDF_1

$ INITIALIZE/QUEUE/DEVICE=SERVER/NOENABLE_GENERIC -
$_          /PROCESSOR=pmdf_process_smb/ON=node:: -
$_          /PROTECTION=(S:RWE, O:RWD, G:R, W:R) PMDF_2

$ INITIALIZE/QUEUE/DEVICE=SERVER/NOENABLE_GENERIC -
$_          /PROCESSOR=pmdf_process_smb/ON=node:: -
$_          /PROTECTION=(S:RWE, O:RWD, G:R, W:R) PMDF_3

$ INITIALIZE/QUEUE/DEVICE=SERVER/NOENABLE_GENERIC -
$_          /PROCESSOR=pmdf_process_smb/ON=node:: -
$_          /PROTECTION=(S:RWE, O:RWD, G:R, W:R) PMDF_4

$ INITIALIZE/QUEUE/DEVICE=SERVER/GENERIC=(PMDF_1, PMDF_2, PMDF_3, PMDF_4) -
$_          /PROTECTION=(S:RWE, O:RWD, G:R, W:R) MAIL$BATCH
```

9.2 Symbiont Option Files

When the first PMDF Process Symbiont server queue is started, then the file pmdf_process_smb.opt in the PMDF table directory is consulted for option settings. If the file does not exist, the symbiont will use internal defaults. The file contains lines of the form

```
option = value
```

[1] Omit the /ON= qualifier if the node is not a member of a cluster.

An exclamation mark, !, introduces a comment line. Blanks and tabs are ignored as well, except within the *value* string, where they are compressed to a single blank.

Note: Particularly in regard to white space, the format of the PMDF Process Symbiont option file differs from the format of other PMDF option files.

In addition to option settings, the file can contain a line consisting of a Process Symbiont queue name enclosed in square-brackets of the form

```
[queue-name]
```

Such a line indicates that option settings following this line are to apply only to the queue named by *queue-name*. Initial option settings that appear before any such queue name tag will apply globally to all queues controlled by the Process Symbiont. Per queue option settings will override global defaults for that queue. Option and queue names are not case sensitive.

A sample Process Symbiont option file is shown in Figure 9–1.

Figure 9–1 Sample Process Symbiont Option File

```
!
! Global defaults for all queues:
!
idle_timeout = 0 05:00:00 ❶
!
! Now queue specific settings:
!
[PMDF_1] ❷   !first queue
lifetime = 0 12:00:00 ❸
!
[PMDF_2] ❹   !second queue
process_priority = 3 ❺
idle_timeout = 0 00:00:01 ❻
```

The key items in the above example are

❶ Set the idle timeout for all queues to 5 hours. This effectively allows PMDF to avoid process creations for PMDF jobs nearly all of the time.

❷ Subsequent options will apply only to the queue named PMDF_1.

❸ Set the life time to 12 hours for jobs in the PMDF_1 queue. This will cause the server process to exit and restart at most every 12 hours.

❹ Subsequent options will apply only to the queue named PMDF_2.

❺ Set the process priority to 3 for jobs in the PMDF_2 queue.

❻ Set the idle timeout to one second for jobs in the PMDF_2 queue. This will cause a single process to be reused only when PMDF has a constant stream of work for this queue. Most of the time there will be no process idle on this queue.

The available options are:

IDLE_TIMEOUT
An OpenVMS delta time which specifies the interval that a server process will wait for another task after it completes a task. If the IDLE_TIMEOUT period expires the server process will be deleted and a new process must be created the next time a task is to be processed. If no IDLE_TIMEOUT is specified, the built-in default of one minute will be used.

PROCESS_PRIORITY
An integer value between 0 and 15 which specifies the OpenVMS process priority for the server task. If no priority is specified, the built-in default of 4 will be used.

LIFETIME
An OpenVMS delta time which specifies the maximum period over which a single server process will be used. If LIFETIME expires for a server process it will be deleted when it finishes the current task. The next task will then cause a new server process to be created. The task process' log file will be closed and a new log file started. If no LIFETIME is specified, the built-in default of 24 hours will be used.

9.3 Restrictions and Limitations

Since OpenVMS does make some distinction between the capabilities of a batch queue and those of a server queue, there are several parameters used on a typical SUBMIT command which will not carry over to the Process Symbiont. Parameters such as /LOG, /NOPRINT, or /WSEXTENT will result in an informational message

```
%JBC-I-ITMREMOVED, meaningless items removed from request
```

This message can generally be ignored. Some PMDF components can produce this message as well, but will nevertheless function properly.

There is currently no way to specify batch job process parameters such as /WSEXTENT on jobs submitted to a PMDF Process Symbiont queue. The working set parameters used will be those specified in the system authorization file for the username under which PMDF jobs are submitted, typically SYSTEM. An informational message will be displayed and the parameter will be ignored.

The Process Symbiont is not intended as a general purpose replacement for OpenVMS batch queues, and is only supported for use with the standard PMDF procedures `master.com`, `post.com`, and `return.com`. Procedures submitted to a process queue will execute under the username of the submitter as would any batch job. The detached server process created for executing jobs queued to the queue *`queue-name`* will unconditionally log output to the file

```
PMDF_LOG:task_server_queue-name.log
```

If the submitting user does not have write access to the PMDF log directory, then the server process will not be able to execute. This effectively limits the Process Symbiont queue to use only by privileged users or the PMDF account, if present, which owns the PMDF log directory. Accordingly, it is suggested that PMDF Process Symbiont queues be protected against user write access to prevent inadvertent loss of user print or batch jobs, (*e.g.*, as shown in the sample INITIALIZE/QUEUE/.../PROTECTION=... commands shown earlier this chapter). For instance, some OpenVMS components or

layered products can detect PMDF Process Symbiont server queues and believe that they are print queues. In particular, the DECwindows print facility can send print jobs to Process server queues since it defaults to using any print or server queue that it sees first in an alphabetic list.

Information is passed from the symbiont to the task server process through the process name. Therefore, a user or system-wide `login.com` which modifies the process name will prevent the task server process from functioning correctly. Such a `login.com` can avoid adverse effects by checking the process mode:

```
$ if F$MODE() .eqs. "OTHER" then exit
```

9.4 Troubleshooting

The PMDF Process Symbiont will send OPCOM messages to terminals enabled for CENTRAL and NETWORK class messages if it detects error or warning conditions. The possible error messages are described below in Section 9.5.

Fatal conditions will result in the Process Symbiont terminating and all execution queues under its control immediately changing to the STOPPED state. In such a situation the OPCOM messages preceding the termination of the symbiont will generally indicate the problem.

Any attempt to start the process symbiont prior to running the PMDF startup command procedure will also cause the symbiont to abort with a fatal error. PMDF_STARTUP must run prior to starting any queues under the control of the process symbiont. See the section on "Post-installation Tasks" in the OpenVMS edition of the *PMDF Installation Guide* for additional restrictions on startup execution order.

Should a detached server process encounter an error condition, it can fail to successfully complete processing of a job. Examination of the server log,

```
PMDF_LOG:task_server_queue-name.log
```

or the task specific log, generally one of

```
PMDF_LOG:post.log
PMDF_LOG:return.log
PMDF_LOG:channel-name_master.log
```

will usually point out the error.

Possible failures to look for are process context or parameters, such as the process name, being altered by a `login.com`, causing the task server to encounter an unexpected condition.

It is also possible for detached server processes to encounter an error condition and terminate abnormally without ever creating a log file. Such terminations will be noted by the Process Symbiont and the process error status sent back to the Job Controller. Setting the execution queue for retention on error with

```
$ SET QUEUE/RETAIN=ERROR queue-name
```

will allow the termination status of the job to be seen by examining the queue after a failure. Examining the OpenVMS accounting records for detached process terminations can also be useful.

9.5 Process Symbiont Errors

Process Symbiont errors will generally result in notification via OPCOM to CENTRAL and NETWORK operator terminals. In addition to standard OpenVMS facility error status messages, the following PMDF specific messages can be encountered.

PMDF-F-PRCSMBFTL, PMDF process symbiont fatal internal error

The symbiont has encountered a fatal condition and will terminate. All execution queues will be stopped. A subsequent message should provide additional information. You should report this error to Process Software technical support, along with all supporting information.

PMDF-F-OPTIONERR, Error reading process symbiont option file on line: !/ !AZ

The symbiont has encountered a syntax error in the option file. Check your option file, `PMDF_TABLE:pmdf_process_smb.opt`. The symbiont will attempt to report the line on which an error was detected.

PMDF-F-BADSTATE, Unexpected state transition, QUEUE= "!AZ", STATE = "!AZ", EVENT = "!AZ"

The symbiont has detected an unexpected sequence of events. You should report this error to Process Software technical support, along with all supporting information.

PMDF-W-PRCSMBWRN, PMDF process symbiont non-fatal warning

The symbiont has encountered a warning condition. Subsequent messages indicate the nature of the warning.

PMDF-W-INVMRNOTIFY, Invalid or unknown Message Router notification message

The PMDF_MR_NOTIFY device has received an invalid message. Notification is ignored. If this condition persists there can be an incompatibility with Message Router.

PMDF-W-MRLSUBERR, Error submitting PMDF-MR master for local mailbox "!AD", status = !UL

The PMDF-MR process could not be started in response to Message Router notification. This can indicate an incorrect PMDF-MR configuration. Additional information from PMDF itself should follow this message.

PMDF-W-MRRSUBERR, Error submitting PMDF-MR master for node !AD, mailbox "!AD", status = !UL

PMDF-MR process could not be started in response to Message Router notification. This can indicate an incorrect PMDF-MR configuration. Additional information from PMDF itself should follow this message.

PMDF-W-DNETDISABL, PMDF_MR_NTFY DECnet object error, remote notify has been disabled

An unrecoverable error has been encountered with the PMDF_MR_NTFY object. The task object will be disabled and will no longer respond to notification requests. This

generally indicates a network problem. A subsequent message should follow with additional information.

PMDF-W-DNETERROR, PMDF_MR_NTFY DECnet object error

A temporary error has been encountered with the PMDF_MR_NTFY object. The task object will continue to respond for notification messages. A subsequent message should follow with additional information.

PMDF-W-DNETSHUT, DECnet network shutdown, remote notify has been disabled

DECnet has been shut down. The task object will no longer respond to notification requests. All Process Symbiont execution queues should be stopped and then restarted in order to restart the symbiont after DECnet is brought back up.

PMDF-W-DNETUNKMSG, Unexpected DECnet control message received, MSGTYPE = !UL

PMDF_MR_NTFY task object has received an unexpected DECnet control message. This indicates an unexpected condition. Please report this to Process Software technical support; be sure to note the MSGTYPE.

10 The PMDF Job Controller (UNIX and Windows)

Note: The PMDF Job Controller is only used on UNIX and Windows platforms. PMDF uses the OpenVMS Job Controller, as well as batch queues or PMDF Process Symbiont queues on OpenVMS.

The PMDF Job Controller is responsible for controlling the flow of messages inside PMDF. When a message is enqueued to a channel, the Job Controller runs the jobs required to deliver the message. Depending on the channel and how heavily the system is loaded, the Job Controller may start a new process, add a thread to an existing process, or verify that a process is already running to deliver the message.

Internally, the Job Controller maintains a set of queues containing process requests. Each message generates a processing request when it is enqueued to a channel, which is placed on the appropriate queue. Each queue has a job limit, which specifies the maximum number of processes that are allowed to run in parallel on that queue. Requests are executed immediately until the job limit is exceeded, at which point they are queued to run as soon as a job is available.

The Job Controller uses an in-memory queue cache database to keep track of message files as they are processed and moved between channels. Normally, every message file in PMDF's on-disk queue area has a corresponding entry in the queue cache database. In extreme cases, the number of messages on disk may exceed the size limit of the in-memory database structure. If this happens, the Job Controller tracks as many message files as it can fit in its database. After enough messages have been processed to free up space inside the database, the Job Controller automatically scans the on-disk queue areas to update its list of message files. If your site frequently experiences heavy message backlogs, the `max_messages` option may be used to tune the number of messages stored in the queue cache database. See Section 10.3 for more information.

Note: Versions of PMDF prior to V6.2-1 have a limited number of requests (specified by the `capacity` keyword) that were stored for each queue. The Job Controller in PMDF V6.2-1 and later stores requests for every message in the queue cache database.

10.1 Job Controller Configuration

During startup, the PMDF Job Controller reads a configuration file that specifies parameters, queues, and channel processing information. This configuration information is specified in `/pmdf/table/job_controller.cnf` (UNIX) or `C:\pmdf\table\job_controller.` (NT). The format of the configuration file is described in Section 10.3. There is also a site-specific configuration file `job_controller.cnf_site`.

If you want to modify the default queue configuration or add additional queues, you can do so by modifying the job controller configuration, and then stopping and restarting the Job Controller with the command:

```
pmdf restart job_controller
```

On NT, you can also restart the Job Controller from the Services screen under the Control Panel.

A new Job Controller process is created, using the new configuration, and receives subsequent requests. The old Job Controller process continues to execute any requests it has queued until they are all finished, and then it exits. Note that you can stop the Job Controller at any time using the command:

```
pmdf shutdown job_controller
```

which gracefully shuts down the Job Controller, allowing any queued requests to finish. On NT, you can also shut down the Job Controller from the Services screen under the Control Panel.

The `DEFAULT` queue in the Job Controller configuration file, by default the only queue, is used for any requests which do not explicitly specify the name of a queue. Processing requests for specific channels can be directed to a specified queue by using the `queue` option followed by the name of the queue. This name must match the name of a defined queue. If the Job Controller does not recognize the requested queue name, the `DEFAULT` queue is used.

Typically, you would add additional queues to the Job Controller configuration if you wanted to separate processing of some channels from that of other channels. For example, you might need to prevent messages sent to a relatively slow channel from blocking processing of messages sent to other channels.

You might also choose to use queues with different characteristics. For example, you might need to control the number of simultaneous requests that some channels are allowed to process. You can do this by creating a new queue with the desired job limit and then use the `queue` option to direct those channels to the new, more appropriate queue.

In addition to the definition of queues, the Job Controller configuration file also contains a table of PMDF channels and the commands that the Job Controller is to use to process requests for each channel. There are two types of requests, termed *master* and *slave*. Typically, a channel's master program is invoked when there is a message stored in a PMDF message queue for the channel. The master program dequeues the message and deivers it. A slave program is invoked to poll a channel and pick up any messsages inbound on that channel. While nearly all PMDF channels have a master program, many do not need a slave program. For example, a channel which handles SMTP over TCP/IP doesn't use a slave program because a network service, the SMTP server, receives incoming SMTP messages upon request by any SMTP client. The SMTP channel's master program is PMDF's SMTP client.

10.2 Default Configuration

PMDF is distributed with a default Job Controller configuration that is suitable for most sites. This default configuration defines a single queue named DEFAULT with a job limit of 4. The DEFAULT queue is used by all PMDF channels which do not specify a queue using the queue option. (In the default configuration, the queue DEFAULT is actually the only queue.)

In addition, the supplied Job Controller configuration file includes channel definitions for all of the supplied and supported PMDF channels.

The Job Controller configuration file is required. If it is not present, or its contents are incorrect, the Job Controller does not start.

There is no need to modify the configuration file unless you choose to add queues, modify queue parameters, modify channel options, or add processing information for locally developed channels. If you do want to make such modifications, you should not alter the Job Controller configuration file itself since it is replaced when you upgrade PMDF and you lose your modifications. Instead, create a job_controller.cnf_site file in the PMDF table directory containing your own definitions. The Job Controller configuration file reads in this site supplied file, if it exists.

10.3 Configuration File Format

The Job Controller configuration file contains lines of the form

option=value

in accordance with the format of PMDF option files.

In addition to option settings, the file can contain a line consisting of a section and value enclosed in square-brackets in the form

[*section-type=value*]

Such a line indicates that option settings following this line are to apply only to the section named by *value*. Initial option settings that appear before any such section tags apply globally to all sections.

Per section option settings override global defaults for that section.

Recognized section types for the Job Controller configuration file are [QUEUE] to define queues and their parameters, and [CHANNEL] to define channel processing information.

A sample Job Controller configuration file on UNIX is shown in Figure 10–1; a sample Job Controller configuration file on NT is shown in Figure 10–2.

Figure 10–1 Sample Job Controller Configuration File on UNIX

```
!
! Global parameters and defaults for all queues
! and channels.
!
TCP_PORT=27442      ❶
SLAVE_COMMAND=NULL  ❷
!
! The DEFAULT queue, which is used for channels that don't
! specify a queue.
!
[QUEUE=DEFAULT]  ❸
JOB_LIMIT=8      ❹
!
! Another queue for some channels to use.
!
[QUEUE=BIGQUEUE]
JOB_LIMIT=16
!
! Definitions for channel processing; each section
! corresponding to a PMDF channel.
!
[CHANNEL=l]      ❺
MASTER_COMMAND=/pmdf/bin/l_master
!
[CHANNEL=tcp_*]
MASTER_COMMAND=/pmdf/bin/tcp_smtp_client
ANON_HOST=0
!
[CHANNEL=cc_*]  ❻
MASTER_COMMAND=/pmdf/bin/lan_master
SLAVE_COMMAND=/pmdf/bin/lan_slave
```

Figure 10–2 Sample Job Controller Configuration File on NT

```
!
! Global parameters and defaults for all queues
! and channels.
!
TCP_PORT=27442      ❶
SLAVE_COMMAND=NULL  ❷
!
! The DEFAULT queue, which is used for channels that don't
! specify a queue.
!
[QUEUE=DEFAULT]  ❸
JOB_LIMIT=8      ❹
!
! Another queue for some channels to use.
!
[QUEUE=BIGQUEUE]
JOB_LIMIT=16
```

Figure 10–2 Cont'd on next page

Figure 10–2 (Cont.) Sample Job Controller Configuration File on NT

```
!
! Definitions for channel processing; each section
! corresponding to a PMDF channel.
!
[CHANNEL=l]      ❺
MASTER_COMMAND=pmdf_exe:msgstore_master
!
[CHANNEL=tcp_*]
MASTER_COMMAND=pmdf_exe:tcp_smtp_client
ANON_HOST=0
!
[CHANNEL=cc_*] ❻
MASTER_COMMAND=pmdf_exe:lan_master
SLAVE_COMMAND=pmdf_exe:lan_slave
```

The key items in the above examples are:

❶ This global option defines the TCP port number on which the Job Controller listens for requests.

❷ Set a default SLAVE_COMMAND for subsequent [CHANNEL] sections.

❸ This [QUEUE] section defines a queue named "DEFAULT". This queue is used by all channels that do not specify a queue name using the queue channel keyword.

❹ Set the JOB_LIMIT for this queue to 8.

❺ This [CHANNEL] section applies to a channel named "l", the PMDF local channel. The only definition required in this section is the MASTER_COMMAND that the Job Controller issues to run this channel. Since no wildcard appears in the channel name, the channel must match exactly.

❻ This [CHANNEL] section applies to any channel whose name begins with "cc_*", i.e., to any PMDF-LAN cc:Mail channel. For this channel, both a MASTER_COMMAND and a SLAVE_COMMAND are necessary. Since this channel name includes a wildcard, it matches any channel whose name begins with "cc_".

Note: The options capacity and udp_port which were valid in Job Controller configuration files prior to PMDF V6.2-1 have been deprecated. They are ignored by PMDF V6.2-1 and later.

Note: Although several Job Controller configuration file options are altered or deprecated in PMDF V6.2-1, existing Job Controller configuration files should still be properly interpreted by PMDF.

The general Job Controller options are:

DEBUG (integer)
An integer value specifying the level of debugging information written to the Job Controller log file in the PMDF log directory. The value of DEBUG is a bit mask specifying what sort of debugging information is requested:

1	Protocol messages between the Job Controller and other PMDF components.
2	Detailed analysis of protocol messages and interactions.
4	Host, queue, and job information.
8	Queue cache database rebuild decisions.
16	Detailed process tracing.
32	Dump each queue on every queue-related action.

Specifying bits 16 or 32 can cause log files to grow very rapidly. Specifying bit 32 will have a noticeable negative impact on PMDF performance.

MAX_MESSAGES (integer)
An integer specifying the number of messages that the Job Controller should provide space for in the in-memory queue cache database. The default value is 100,000, which is sufficient for most sites. If your site regularly experiences large message backlogs, you may wish to increase the value of this option. In the case of a message backlog overflowing the queue cache database, PMDF automatically scans the on-disk queue areas and updates the database as soon as space is available. Note that messages are not lost when the queue cache database fills - they are simply left in the on-disk queue area.

SECRET (integer)
An integer number containing a shared secret to be used for checksum functions, with a default value of 0. The Job Controller requires all incoming requests to be authenticated with a checksum code. This prevents a malicious site from sending malformed requests to the Job Controller.

TCP_PORT (integer)
An integer number indicating the TCP port on which the Job Controller should listen for incoming requests. The default is port 27442. This option shouldn't be changed unless it conflicts with another application on your system.

The options valid in a [QUEUE] section are:

JOB_LIMIT (integer)
The maximum number of jobs (processes) that a queue may run in parallel to handle requests. If there are more requests in the queue than there are available jobs, they are held until a job becomes available to process them. The JOB_LIMIT applies to each queue individually - the maximum total number of jobs is the sum of the JOB_LIMIT parameters for all queues. If this parameter is set outside of a [QUEUE] section, it is used as the default by any queue that doesn't specify a JOB_LIMIT. The default value is 1.

The options valid in a [CHANNEL] section are:

ANON_HOST (0 or 1)

This option is only useful for tcp_* channels. For all other channels, it should be left to the default value of 1.

For TCP channels, when anon_host is set to the default value of 0, this causes the Job Controller to group together messages that are going to be sent to the same remote host (based on the domain name in the envelope TO address), and then treat them as a group. This is the most efficient processing method for the standard tcp_local or tcp_internal channels, which are expected to be sending mail to lots of different remote domain names, with a small number going to each domain name.

Setting the anon_host option to 1 for TCP channels is recommended for specialty channels (that can be configured by the system manager on a site-specific basis) which are set up to send all of its messages to a single remote system. Because this setting allows all of the threads in all of the SMTP client jobs for that channel to be used simultaneously to handle all of the messages. (With anon_host=0, one thread is dedicated to each remote domain name.)

MASTER_COMMAND (file specification)

Specifies the full path to the command to be executed by the Job Controller to run the channel and dequeue outbound messages in the channel. Some PMDF channels do not have a MASTER_COMMAND. If that is the case, the reserved value NULL should be specified. If this parameter is set outside of a [CHANNEL] section, it is used as the default by any channel that doesn't specify a MASTER_COMMAND.

MAX_AGE (integer)

Specifies the maximum lifetime for a channel master job in seconds. When a channel master job has been running for this period of time, it is shut down. This prevents master jobs for heavily used channels from monopolizing the available job slots, and gives lesser-used channels a chance to run. If no value is specified, the default value of 1800 (30 minutes) is used.

MAX_CONNS (integer)

For TCP channels only, in addition to the MAX_AGE parameter, the lifetime of a tcp_xxx channel master job is limited by the number of times it can check with the Job Controller to see if there are any queued requests. If this parameter is not specified for a channel, the default value of 300 is used.

QUEUE (queue name)

By default, all channels use the DEFAULT queue. If you have defined additional queues, use the QUEUE option to direct certain channels to run in certain queues.

SLAVE_COMMAND (file specification)

Specifies the full path to the command to be executed by the Job Controller to run the channel and poll for any messages inbound on the channel. Most PMDF channels do not have a SLAVE_COMMAND. If that is the case, the reserved value NULL should be specified. If this parameter is set outside of a [CHANNEL] section, it is used as the default by any channel that doesn't specify a SLAVE_COMMAND.

10.4 Adding Additional Queues

The Job Controller creates and manages channel jobs for delivering messages. Each set of channel jobs is part of an internal queue. The queue provides an area where a set of channel jobs can run without contention for resources with other channel jobs outside of the queue. The number of jobs that can run inside a particular queue is set with the JOB_LIMIT keyword. For example, if you have a CONV queue defined with a JOB_LIMIT of 10, then only 10 processes can run in that queue at any given time.

Usage of the conversion channel for virus scanning or stripping "dangerous" attached files is a common situation where sites would want to create additional queues. Virus scanning a message requires a significant amount of time, and conversion channel jobs can quickly fill every available job slot at a busy site. Delivery channel jobs such as the l (local) and msgstore channels are only allowed to run infrequently, causing large delays in message delivery.

A solution to this problem is to define a queue named CONV in the Job Controller configuration file (job_controller.cnf) that is dedicated to handling conversion channel jobs:

```
[QUEUE=CONV]
JOB_LIMIT=10
```

The conversion channel is told to run its channel jobs inside the new queue with the queue keyword. For example:

```
[CHANNEL=conversion]
master_command=/pmdf/bin/conversion
queue=conv
```

This allows up to 10 conversion channel jobs to run in parallel, while leaving all of the job slots in the other queues available for other channel jobs.

10.5 Checking that the PMDF Job Controller is Running

You can verify that the PMDF Job Controller is running with the command pmdf process on either UNIX or NT, or by checking under the Control Panel under Services for the Job Controller on NT. On UNIX, you should see output similar to that shown below, perhaps with additional jobs present if your system is currently processing messages.

```
# pmdf process
   USER     PID  S   VSZ      RSS      STIME      TIME     COMMAND
   pmdf    4396  S  16080    8064    14:09:13    0:01    <IMAP_SERVER>
   pmdf    4393  S  15968    7816    14:09:13    0:00    <POP_SERVER>
   pmdf    4395  S  15624    7568    14:09:13    0:00    <SMTP>
   pmdf    4390  S  15080    6680    14:09:12    0:00    /pmdf/bin/dispatcher
   pmdf    4392  S  15600    7448    14:09:13    0:01    <HTTP>
   root    4394  S  15928    7736    14:09:13    0:00    <POPPASSD>
   pmdf    4374  S  12960    2456    14:09:12    0:00    /pmdf/bin/job_controller
```

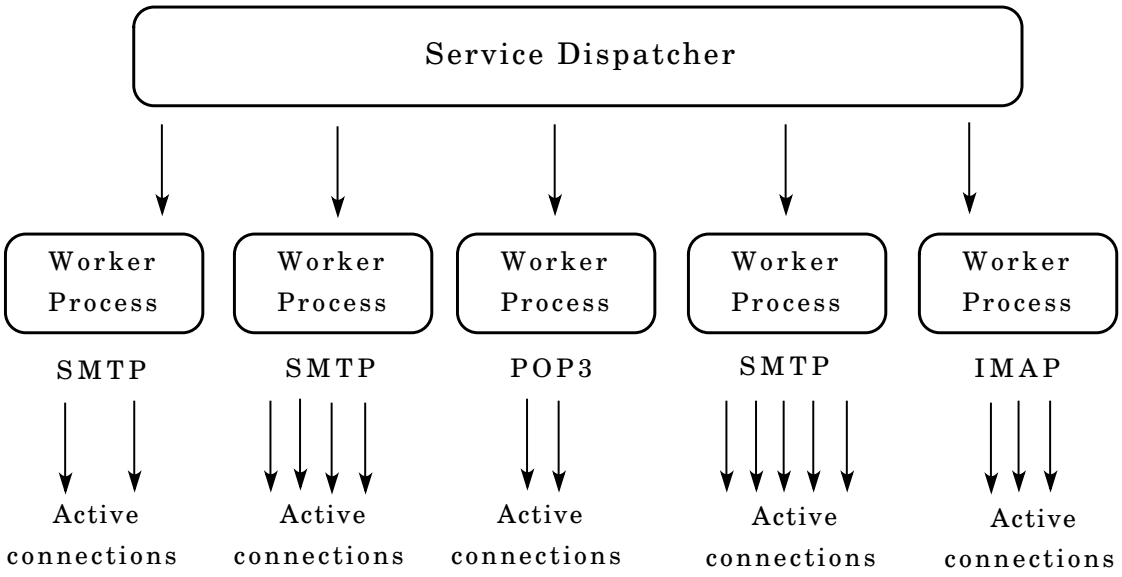
11 The PMDF Multithreaded Service Dispatcher

The PMDF multithreaded Service Dispatcher is a multithreaded connection dispatching agent that permits multiple multithreaded servers to share responsibility for a given service. When using the Service Dispatcher, it is possible to have several multithreaded SMTP, POP3, and IMAP servers running concurrently. In addition to having multiple servers for a single service, each server can have one or more active connections.

11.1 Operation of the Service Dispatcher

The Service Dispatcher works by acting as a central receiver for the TCP ports listed in its configuration. For each defined service, the PMDF Service Dispatcher can create one or more Worker Processes that will actually handle the connections after they've been established. A schematic of a Service Dispatcher and its Worker Processes is shown in Figure 11–1.

Figure 11–1 The Service Dispatcher and Its Worker Processes



In general, when the Service Dispatcher receives a connection for a defined TCP port, it checks its pool of available Worker Processes and chooses the best candidate for the new connection. If no suitable candidate is available and the configuration permits it, the Service Dispatcher can create a new Worker Process to handle this and subsequent connections. The Service Dispatcher can also proactively create a new Worker

Process in expectation of future incoming connections. There are several configuration options which can be used to tune the PMDF Service Dispatcher's control of its various services, and in particular, to control the number of Worker Processes and the number of connections each Worker Process handles; see Section 11.1.1 and Section 11.3.

11.1.1 Creation and Expiration of Worker Processes

There are automatic housekeeping facilities within the Service Dispatcher to control the creation of new and expiration of old or idle Worker Processes. The basic options that control the Service Dispatcher's behavior in this respect are `MIN_PROCS` and `MAX_PROCS`. `MIN_PROCS` provides a guaranteed level of service by having a number of Worker Processes ready and waiting for incoming connections. `MAX_PROCS`, on the other hand, sets an upper limit on how many Worker Processes can be concurrently active for the given service.

Since it is possible that a currently running Worker Process might not be able to accept any connections either because it is already handling the maximum number of connections of which it is capable or because the process has been scheduled for termination, the Service Dispatcher can create additional processes to assist with future connections.

The `MIN_CONNS` and `MAX_CONNS` options provide a mechanism to help you distribute the connections among your Worker Processes. `MIN_CONNS` specifies the number of connections that flags a Worker Process as "busy enough" while MAX_CONNS specifies the "busiest" that a Worker Process can be.

In general, the Service Dispatcher will create a new Worker Process when the current number of Worker Processes is less than `MIN_PROCS` or when all existing Worker Processes are "busy enough" (the number of currently active connections each has is at least `MIN_CONNS`).

Note that if a Worker Process is killed unexpectedly, *e.g.*, by the OpenVMS DCL `STOP/ID` command or the UNIX `kill` command, the Service Dispatcher will still create new Worker Processes as new connections come in.

11.2 Required Software Versions

On OpenVMS, the Service Dispatcher requires as the underlying TCP/IP package, any one of MultiNet V3.5A or later with all UCXDRIVER patches (upgrading to MultiNet V4.0B, which requires no known patches as of this printing, is recommended), Pathway V2.5.x with all patches (in particular C82195) or later, TCPware V5.0 or later, or TCP/IP Services for OpenVMS V4.0 or later (with UCX V4.1 requiring upgrading to at least ECO 5).

On OpenVMS, note that the Service Dispatcher requires that your TCP/IP package provide UCX emulation; if your TCP/IP package is not currently configured for UCX emulation, you must enable UCX emulation.

Your TCP/IP package must be configured to include a loopback adaptor; that is, your TCP/IP package must recognize the IP address 127.0.0.1 as itself.

11.3 The Dispatcher Configuration File

The Service Dispatcher requires a configuration file to tell it what services to handle and to set various options for those services. A framework Dispatcher configuration file is shipped with PMDF, referenced by the `PMDF_DISPATCHER_CONFIG_MAIN` logical name (OpenVMS), or PMDF tailor file option (UNIX), or PMDF Tailor Registry entry (NT), and hence normally `dispatcher_main.cnf` located in the PMDF table directory; normally this framework configuration file should not be site modified. (In particular, on UNIX and NT the framework Dispatcher configuration file defines some internal PMDF services, `PWCHECK` and `PFILE`, whose service definitions should *not* be modified.)

The framework Dispatcher configuration file reads in the site-specific Service Dispatcher configuration file which *is* intended to be site created and site modified, located via the `PMDF_DISPATCHER_CONFIG` logical name (OpenVMS), or PMDF tailor file option (UNIX), or PMDF Tailor Registry entry (NT) and hence is usually `dispatcher.cnf` located in the PMDF table directory. The command line PMDF Service Dispatcher configuration utility, `pmdf configure dispatcher` (UNIX and VMS), or the PMDF web-based configuration utility (all platforms), should be used to create an initial configuration file; see the appropriate edition of the *PMDF Installation Guide* for instructions and an example of using this utility.

11.3.1 Configuration File Format

The Service Dispatcher configuration file format is similar to the format of other PMDF option files. Lines specifying options have the form

option=value

where *option* is the name of an option and *value* is the string or integer to which to set the option. If the option accepts an integer value, *value*, a base can be specified using notation of the form *b%v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*. Such option specifications are grouped into sections corresponding to the service to which the following option settings apply via lines of the form

SERVICE=*service-name*

where *service-name* is the name of a service. Initial option specifications that appear before any such section tag apply globally to all sections. A `[SERVICE=DISPATCHER]` section can be used to specify options for the Dispatcher itself.

A sample configuration file for an OpenVMS system is shown in Figure 11–2; a sample configuration file for a UNIX system is shown in Figure 11–3; a sample configuration file for an NT system is shown in Figure 11–4.

Figure 11–2 Sample Service Dispatcher Configuration File on OpenVMS, dispatcher.cnf

```
! The first set of options, listed without a [SERVICE=xxx]
! header, are the default options that will be applied to all
! services.
!
! Global defaults
!
MIN_PROCS=1
MAX_PROCS=5
MIN_CONNS=3
MAX_CONNS=10
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=300
MAX_IDLE_TIME=600
!
! Other files can be included using the "<" operator as below
!
<PMDF_TABLE:dispatcher_site.cnf
!
! Now, define the services available to the Dispatcher
!
! multithreaded SMTP server
!
[SERVICE=SMTP]
PORT=25
IMAGE=PMDF_EXE:tcp_smtp_server.exe
LOGFILE=PMDF_LOG:tcp_smtp_server.log
!
! HTTP server
!
[SERVICE=HTTP]
PORT=7633
IMAGE=PMDF_EXE:http_server.exe
LOGFILE=PMDF_LOG:http_server.log
!
! POP3 server
!
[SERVICE=POP3]
PORT=110
IMAGE=PMDF_EXE:pop3d.exe
LOGFILE=PMDF_LOG:pop3d.log
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
IMAGE=PMDF_EXE:imapd.exe
LOGFILE=PMDF_LOG:imapd.log
```

Figure 11–3 Sample Service Dispatcher Configuration File on UNIX, `dispatcher.cnf`

```
! The first set of options, listed without a [SERVICE=xxx]
! header, are the default options that will be applied to all
! services.
!
MIN_PROCS=1
MAX_PROCS=5
MIN_CONNS=3
MAX_CONNS=10
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=300
MAX_IDLE_TIME=600
!
! Other files can be included using the "<" operator as below
!
</pmdf/table/dispatcher_site.cnf
!
! Now, define the services available to Dispatcher
!
[SERVICE=SMTP]
PORT=25
IMAGE=/pmdf/bin/tcp_smtp_server
LOGFILE=/pmdf/log/tcp_smtp_server.log
!
! HTTP server
!
[SERVICE=HTTP]
PORT=7633
IMAGE=/pmdf/bin/http_server
LOGFILE=/pmdf/bin/http_server.log
!
[SERVICE=POP3]
PORT=110
IMAGE=/pmdf/bin/pop3d
LOGFILE=/pmdf/log/pop3d.log
!
[SERVICE=IMAP]
PORT=143
IMAGE=/pmdf/bin/imapd
LOGFILE=/pmdf/bin/imapd.log
```

11.3.2 Available Options

The available Dispatcher configuration file options are:

ASTLM, BIOLM, BYTLM, CPULM, DIOLM, ENQLM, FILLM, JTQUOTA, PGFLQUOTA, PRCLM, TQELM, WSDEFAULT, WSEXTENT, WSQUOTA (integer; OpenVMS only)
On OpenVMS, use the specified process quota. The default values are:

Figure 11–4 Sample Service Dispatcher Configuration File on NT, `dispatcher.cnf`

```
! The first set of options, listed without a [SERVICE=xxx]
! header, are the default options that will be applied to all
! services.
!
MIN_PROCS=1
MAX_PROCS=5
MIN_CONNS=3
MAX_CONNS=10
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=300
MAX_IDLE_TIME=600
!
! Other files can be included using the "<" operator as below
!
<C:\pmdf\table\dispatcher_site.cnf
!
! Now, define the services available to Dispatcher
!
[SERVICE=SMTP]
PORT=25
IMAGE=C:\pmdf\bin\tcp_smtp_server
LOGFILE=C:\pmdf\log\tcp_smtp_server.log
!
! HTTP server
!
[SERVICE=HTTP]
PORT=7633
IMAGE=C:\pmdf\bin\http_server
LOGFILE=C:\pmdf\bin\http_server.log
!
[SERVICE=POP3]
PORT=110
IMAGE=C:\pmdf\bin\pop3d
LOGFILE=C:\pmdf\log\pop3d.log
!
[SERVICE=IMAP]
PORT=143
IMAGE=C:\pmdf\bin\imapd
LOGFILE=C:\pmdf\bin\imapd.log
```

Option	Default value
ASTLM	500
BIOLM	100
BYTLM	200000
CPULM	0
DIOLM	100
ENQLM	0
FILLM	0
JTQUOTA	0
PGFLQUOTA	81920
PRCLM	0
TQELM	500
WSDEFAULT	0
WSEXTENT	0
WSQUOTA	0

A default value of 0 means to use the corresponding PQL_D*SYSGEN parameter value.

For POP and IMAP services, the BYTLM, ENQLM, FILLM, and PGFLQUOTA options are particularly relevant. A general recommendation for sites using a POP or IMAP server is to set the Dispatcher option BYTLM in the POP or IMAP service section, respectively, of the Dispatcher configuration file according to the formula:

$$\mathrm{BYTLM} > 5120 * \sum_{services} \mathrm{MAX\_PROCS} + 1024 * \sum_{services} (\mathrm{MAX\_PROCS} * \mathrm{MAX\_CONNS}).$$

Or if UCX_HOLD=0 is set, the need for BYTLM is instead only

$$\mathrm{BYTLM} > 5120 * \sum_{services} \mathrm{MAX\_PROCS}$$

BACKLOG (integer)
This option controls the depth of the TCP backlog queue for the socket. The default value for each service is MAX_CONNS*MAX_PROCS for that service (with a minimum value of 5). On OpenVMS, the maximum value is 255; attempts to set higher values will be treated as a value of 255. This option should not be set higher than the underlying TCP/IP kernel supports.

DNS_VERIFY_DOMAIN (host name or IP address)
Various groups maintain information about spam sources or open relay sites and some sites like to check incoming IP connections against the lists maintained by such groups. This option specifies the host name or IP address of source against which to check incoming connections. You can have up to five DNS_VERIFY_DOMAIN options for each service. (Note that SMTP is typically the only service for which such checks make sense.) For example:

```
[SERVICE=SMTP]
PORT=25
DNS_VERIFY_DOMAIN=relays.mail-abuse.org
DNS_VERIFY_DOMAIN=dialups.mail-abuse.org
```

If this option is enabled on a well-known port (25, 110, or 143), then a standard message such as the one below will be sent before the connection is closed:

```
500 5.7.1 access_control: host 192.168.51.32 found on DNS list and rejected
```

If you want PMDF to log such rejections, you can set the 24th bit of the Dispatcher debugging DEBUG option, DEBUG=16%1000000, to cause logging of the rejections to the `dispatcher.log` file; see Section 11.6. Such `dispatcher.log` entries will take the form:

```
access_control: host a.b.c.d found on DNS list and rejected
```

ENABLE_RBL (0 or 1)

Specifying `ENABLE_RBL=1` causes the Dispatcher to compare incoming connections to the "Black Hole" list at mail-abuse.org.

Note: Setting ENABLE_RBL to 1 is the same as using the option DNS_VERIFY_DOMAIN set to blackholes.mail-abuse.org. The ENABLE_RBL option has been obsoleted by the DNS_VERIFY_DOMAIN option.

For example, if the Dispatcher receives a connection from 192.168.51.32, then it will attempt to obtain the IP address for the hostname 32.51.168.192.blackholes.mail-abuse.org. If the query is successful, the connection will be closed rather than handed off to a worker process.

If this option is enabled on a well-known port (25, 110, or 143), then a standard message such as the one below will be sent before the connection is closed:

```
500 5.7.1 access_control: host 192.168.51.32 found on DNS list and rejected
```

If you want PMDF to log such rejections, you can set the 24th bit of the Dispatcher debugging `DEBUG` option, `DEBUG=16%1000000`, to cause logging of the rejections to the `dispatcher.log` file; see Section 11.6. Such `dispatcher.log` entries will take the form:

```
access_control: host a.b.c.d found on DNS list and rejected
```

GROUP (string; UNIX only)
USER (string; UNIX only)

These options control under what user id and group id the service runs. Via these options, the Dispatcher can give services the access they need to function properly. The IMAP, POP3, and POPPASSD servers should be set

```
USER=root
```

These options *should not be set* except to those values and for those services where Process Software specifically directs their use.

HISTORICAL_TIME (integer)

The `HISTORICAL_TIME` option controls how long (in seconds) expired connections (those that have been closed) and processes (those that have exited) remain listed for statistical purposes. The default value is 120 seconds; *i.e.*, two minutes. Note that the setting of this

option affects the amount of virtual memory that the Dispatcher requires; for instance, on OpenVMS, busy sites that want to increase the HISTORICAL_TIME setting can also need to increase the PGFLQUOTA option setting for the Dispatcher service itself in a [SERVICE=DISPATCHER] section.

IMAGE (file specification)

This is the image that will be run by Worker Processes when created by the Service Dispatcher. Note that the specified image should be one designed to be controlled by the Service Dispatcher.

INTERFACE_ADDRESS (IP address)

The INTERFACE_ADDRESS option can be used to specify the IP address interface to which the Dispatcher service should bind. By default, the Dispatcher binds to all IP addresses. But for systems having multiple network interfaces each with its own IP address, it can be useful to bind different services to the different interfaces. Note that if INTERFACE_ADDRESS is specified for a service, then that is the *only* interface IP address to which that Dispatcher service will bind. Only one such explicit interface IP address can be specified for a particular service (though other similar Dispatcher services can be defined for other interface IP addresses).

Note that the interfaceaddress channel keyword, Section 2.3.4.37, provides the complementary capability for specifying which interface address a TCP/IP channel uses for outgoing connections and messages.

LOGFILE (file specification)

Specifying this option for a service causes the Service Dispatcher to direct output for corresponding Worker Processes to the specified file. The log file can include the system's name (SCSNODE on VMS, if available, or the first part of the Internet hostname) by including the %s token. For instance,

```
[SERVICE=DISPATCHER]
LOGFILE=PMDF_LOG:dispatcher-%s.log
```

MAX_CONNS (integer)

This option specifies the maximum number of concurrent connections handled by a single server process (Worker Process). When the maximum number of concurrent sessions is reached, the server process stops listening for new connections. When all currently open connections are closed the original server will exit. The default value for this option is 10. On OpenVMS, the maximum possible value for this option is 31, where here the limit is the number of threads supported by OpenVMS's callable MAIL. On other platforms, the maximum possible value for this option is 50.

For services where the server image is not multithreaded, this option must be set to 1.

In contrast, servers such as the PMDF SMTP server, POP3 server, or IMAP server are multi-threaded, and therefore capable of handling multiple clients. For such multithreaded servers, the choice of setting for this option is mainly a performance issue relating to the number of processes and the size of the process virtual address space. Setting MAX_CONNS to higher values allows more connections, but at the potential cost of decreased performance for each individual connection. If it is set to 1, then for every incoming client connection, only one server process will be used. When the client shuts down, the server process will also exit. Note that this value times the MAX_PROCS value controls the maximum number of simultaneous connections that can be accepted.

MAX_HANDOFFS (integer)

This option specifies the maximum number of concurrent asynchronous handoffs in progress that the Dispatcher will allow for newly established TCP/IP connections to a service port. The default value is 5.

MAX_IDLE_TIME (integer)

When a Worker Process has had no active connections for this period, it will be eligible for being shut down. Note that this option is only effective if there are more than the value of `MIN_PROCS` Worker Processes currently in the Service Dispatcher's pool for this service.

MAX_LIFE_CONNS (integer)

As part of the Service Dispatcher's ability to perform Worker Process housekeeping, this option requests that Worker Processes only be kept around for the specified number of connections. After a Worker Process has handled the specified number of connections, it is subject to being shut down. The global default value is 300.

For instance, when specified in a POP or IMAP service section, this is the number of total connections the POP3 or IMAP server is able to accept before being restarted. This is different from the `MAX_CONNS` option, which limits the number of concurrent connections. On OpenVMS when serving out the native VMS MAIL mailbox it is recommended to set this to no more than 100; otherwise problems due to memory leaks in callable MAIL can be encountered.

MAX_LIFE_TIME (integer)

As part of the Service Dispatcher's ability to perform Worker Process housekeeping, this option requests that Worker Processes only be kept around for the specified number of seconds. When a Worker Process is created, a countdown timer is set to the specified number of seconds. When the countdown time has expired, the Worker Process is subject to being shut down.

MAX_PROCS (integer)

This option controls the maximum number of Worker Processes that will be created for this service. Thus this value times `MAX_CONNS` thus specifies the maximum number of simultaneous connections that can be handled.

MAX_SHUTDOWN (integer)

In order to provide a minimum availability for the service, the Service Dispatcher will not shut down Worker Processes that might otherwise be eligible to be shut down if shutting them down would result in having more than `MAX_SHUTDOWN` Worker Processes for the service in the shutting down state. This means that processes that can be eligible to be shut down can continue running until a shutdown "slot" is available. Having this option set to about half of the value of the `MAX_PROCS` option is usually appropriate.

MIN_CONNS (integer)

The Service Dispatcher attempts to distribute connections evenly across its pool of currently available Worker Processes. The Service Dispatcher uses this value to determine the minimum number of connections that each Worker Process must have before there will be any consideration of adding new Worker Processes to the pool.

MIN_PROCS (integer)

This option determines the minimum number of Worker Processes that will be created by the Service Dispatcher for the current service. Upon initialization, the Service Dispatcher will create this many detached processes to start its pool. When a process is shut down, the Service Dispatcher will ensure that there are at least this many available processes in the pool for this service.

PARAMETER

The interpretation and allowed values for the PARAMETER option are service specific. In the case of an SMTP service, the PARAMETER option can be set to CHANNEL=*channelname*, to associate a default TCP/IP channel with the port for that SMTP service. For instance,

```
[SERVICE=SMTP]
PORT=25
...
PARAMETER=CHANNEL=tcp_incoming
```

Note that this can be useful if you want to run SMTP servers on multiple ports – perhaps because your internal POP and IMAP clients have been configured to use a port other than the normal port 25, thus separating their message traffic from incoming SMTP messages from external SMTP hosts—and if you want to associate different TCP/IP channels with the different port numbers.

For an IMAP server or Web500 server, the PARAMETER option can be set to CONFIG_FILE=*filename*, to tell the server to use the specified file as its configuration file. Note that this can be useful if you want to run such servers on multiple ports with the different servers having different configuration files.

PORT (integer or comma separated list of integers)

Specifies the TCP port(s) on which the Service Dispatcher will listen for incoming connections for the current service. Connections made to this port or these ports will be transferred to one of the Worker Processes created for this service. The default is 25 for SMTP, 110 for POP3, and 143 for IMAP. Specifying PORT=0 has the effect of disabling the current service.

PRIORITY (integer; OpenVMS only)

On OpenVMS, the PRIORITY option can be used to control the priority at which the Worker Processes will run.

STACKSIZE (integer > 0)

Specifies a minimum per-thread stack size. Various components can have their own minimum values; the larger of an explicitly specified STACKSIZE option value and the component's own internal minimum will be used.

TLS_CERTIFICATE (comma separated list of file-specs)

This option specifies a pair of files in which a TLS certificate can be found. The default, if this option is not specified, is to look for a certificate in the server-pub.pem and server-priv.pem files stored in the PMDF table directory. Up to five instances of this option can be specified, which can be particularly useful for sites that want to have and use multiple certificates; for instance:

```
TLS_CERTIFICATE=/pmdf/table/server-pub.pem,/pmdf/table/server-priv.pem
TLS_CERTIFICATE=/pmdf/table/server-smtp-pub.pem,/pmdf/table/server-smtp-priv.pem
```

Note: The TLS_CERTIFICATE option is only available for PMDF-TLS sites.

TLS_PORT (integer or comma separated list of integers)

Specifies the TCP port(s) on which the Service Dispatcher will listen for incoming TLS connections for the current service. Connections made to this port or these ports will automatically negotiate TLS use and be transferred to one of the Worker Processes created for this service. See also Section 15.2.2.1.

Note: The TLS_PORT option is only available for PMDF-TLS sites.

UCX_HOLD (0 or 1; OpenVMS only)

On OpenVMS, the `UCX_HOLD` option controls whether the Dispatcher continues to hold an assigned I/O channel for each connection for that service while the connnection is being handled by a worker process. The default value, `UCX_HOLD=1`, is required for some TCP/IP packages, such as DEC TCP/IP Services for OpenVMS (UCX). Keeping such connections open requires that your system have a sufficient `CHANNELCNT SYSGEN` parameter.

With some TCP/IP packages, *e.g.*, MultiNet or TCPware, you can want to set this option to 0, allowing the Dispatcher to deassign the I/O channel after the connection has been handed off to a worker process, and thereby mitigating the need to increase the `CHANNELCNT` value.

WP_TIMEOUT (integer)

This option specifies how many seconds the Dispatcher should wait for a Worker Process to start, before deciding that the Worker Process must be dead and trying to start another one. The default is 30. The value should not be set higher than 60.

11.4 Controlling the Service Dispatcher

The Service Dispatcher is a single resident process which starts and shuts down Worker Processes for various services as needed. The Service Dispatcher process can be started with the command

```
pmdf startup dispatcher
```

Note: The command to start up the Dispatcher subsumes and makes obsolete any other PMDF STARTUP (OpenVMS) or `pmdf startup` (UNIX) command that was used previously in PMDF V5.0 to start up a component of PMDF that the Service Dispatcher has now been configured to manage. Specifically, you should no longer use PMDF STARTUP SMTP, POP3, or IMAP (OpenVMS) or `pmdf startup smtp`, `pop3`, or `imap` (UNIX). An attempt to execute any of the obsoleted commands will cause PMDF to issue a warning.

To shut down the Service Dispatcher, use the command

```
pmdf shutdown dispatcher
```

What happens with the Worker Processes when the Service Dispatcher is shut down depends upon the underlying TCP/IP package.

For instance, on OpenVMS with Multinet as the TCP/IP package shutting down the Service Dispatcher will not terminate any currently active connections served by any Worker Processes. Rather, the command requests that the Service Dispatcher itself terminate, meaning that new connections will no longer be accepted and then assigned to Worker Processes, and that Worker Processes will automatically terminate as they finish previously assigned tasks and become idle.

On OpenVMS with UCX as the TCP/IP package however, shutting down the Service Dispatcher will also cause the Worker Processes to immediately abort their connections.

If you modify your PMDF configuration or options that apply to the Service Dispatcher, you must restart the Service Dispatcher so that the new configuration or options will take effect. Use the command

pmdf restart dispatcher

Restarting the Service Dispatcher has the effect of shutting down the currently running Service Dispatcher and then immediately starting a new one.

Individual services can be restarted only on OpenVMS or UNIX platforms. Use the command

pmdf restart *service-name*

where *service-name* is a service defined in the Service Dispatcher configuration file when the Service Dispatcher was started; for instance, when using the sample Service Dispatcher configuration file, SMTP, POP3, or IMAP (OpenVMS) or `smtp`, `pop3`, or `imap` (UNIX). These commands signal the Service Dispatcher to restart only that component that was specified.

When receiving a command to restart a particular server, the Service Dispatcher first shuts down that service and then immediately starts a new service (re-reading the Service Dispatcher configuration file), creating new Worker Processes as requested.

Individual services can be shut down only on OpenVMS and UNIX platforms. Use the command

pmdf shutdown *service-name*

where *service-name* is a service defined in the Service Dispatcher configuration file when the Service Dispatcher was started.

When receiving a command to shut down a particular service, the Service Dispatcher stops accepting new connections for that service and terminates the Worker Processes for that service when they become idle.

Note: If you shut down a particular service, you will need to restart (or shut down and start up) the Service Dispatcher itself in order to get the service started up again.

11.5 Connection Access Control

The PMDF Service Dispatcher is able to selectively accept or reject incoming SMTP connections based on IP address and port number. At Dispatcher startup time, the Dispatcher will look for a mapping table named `PORT_ACCESS`. If present, the Dispatcher will format connection information in the form:

`TCP|`*server-address*`|`*server-port*`|`*client-address*`|`*client-port*

and try to match against all `PORT_ACCESS` mapping entries. If the result of the mapping contains `$N` or `$F`, the connection will be immediately closed. Any other result of the mapping indicates that the connection is to be accepted. `$N` or `$F` can optionally be followed by a rejection message. If present, the message will be sent back down the

connection just prior to closure. Note that a CRLF terminator will be appended to the string before it is sent back down the connection.

The flag $< followed by an optional string causes PMDF to send the string as an OPCOM broadcast (OpenVMS) or to syslog (UNIX) or to the event log (NT) if access is rejected. If bit 1 of the LOG_CONNECTION PMDF option is set and the $N flag is set so that the connection is rejected, then also specifying the $T flag will cause a "T" entry to be written to the connection log. If bit 4 of the LOG_CONNECTION PMDF option is set, then site-supplied text can be provided in the PORT_ACCESS entry to include in the "C" connection log entries entries; to specify such text, include two vertical bar characters in the right hand side of the entry, followed by the desired text. See Table 11–1 for a summary of the available flags.

Table 11–1 PORT_ACCESS mapping flags

Flag	Description
$Y	Allow access
	Flags with arguments, in argument reading order†
$<*string*	Send *string* as an OPCOM broadcast (OpenVMS) or to syslog (UNIX) or to the event log (NT) if access is rejected
$N*string*	Reject access with the optional error text *string*
$F*string*	Synonym for $N*string*, *i.e.*, reject access with the optional error text *string*
$T*text*	If bit 1 of the LOG_CONNECTION PMDF option is set and the $N flag is set so that the connection is rejected, then $T causes a "T" entry to be written to the connection log; the optional text *text* (which must appear subsequent to two vertical bar characters) can be included in the connection log entry

†To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

For example, the following mapping will only accept SMTP connections (to port 25, the normal SMTP port) from a single network, except for a particular host singled out for rejection without explanatory text:

```
PORT_ACCESS
  TCP|*|25|192.168.10.70|*      $N500
  TCP|*|25|192.168.10.*|*       $Y
  TCP|*|25|*|*                  $N500$ Bzzzzzzzzt$ thank$ you$ for$ playing.
```

Note that you will need to restart the Dispatcher after making any changes to the PORT_ACCESS mapping table so that the Dispatcher will see the changes. (And if you're using a compiled PMDF configuration, you'll first need to recompile your configuration to get the change incorporated into the compiled configuration.)

The PORT_ACCESS mapping table is specifically intended for performing IP number based rejections; for more general control at the email address level, the SEND_ACCESS or MAIL_ACCESS mapping table, as described in Section 16.1, can be more appropriate.

11.6 Debugging and Log Files

Service Dispatcher error and debugging output (if enabled) are written to the file `dispatcher.log` in the PMDF log directory.

Debugging output can be enabled using the option `DEBUG` in the Dispatcher configuration file, or on a per-process level, via the `PMDF_DISPATCHER_DEBUG` logical (OpenVMS) or environment variable (UNIX or NT).

The `DEBUG` option or `PMDF_DISPATCHER_DEBUG` logical name (OpenVMS) or environment variable (UNIX or NT) defines a 32-bit debug mask in hexadecimal. Enabling all debugging is done by setting the option to `-1`, or by defining the logical or environment variable system-wide to the value `FFFFFFFF`. The actual meaning of each bit is described in Table 11–2.

Table 11–2 Dispatcher debugging bits

Bit	Hexadecimal value	Decimal value	Usage
0	x00001	1	Basic Service Dispatcher main module debugging
1	x00002	2	Extra Service Dispatcher main module debugging
2	x00004	4	Service Dispatcher configuration file logging
3	x00008	8	Basic Service Dispatcher miscellaneous debugging
4	x00010	16	Basic service debugging
5	x00020	32	Extra service debugging
6	x00040	64	Process related service debugging
7	x	128	Not used
8	x00100	256	Basic Service Dispatcher and process communication debugging
9	x00200	512	Extra Service Dispatcher and process communication debugging
10	x00400	1024	Packet level communication debugging
11	x00800	2048	Not used
12	x01000	4096	Basic Worker Process debugging
13	x02000	8192	Extra Worker Process debugging
14	x04000	16384	Additional Worker Process debugging, particularly connection handoffs
15	x08000	32768	Not used
16	x10000	65536	Basic Worker Process to Service Dispatcher I/O debugging
17	x20000	131072	Extra Worker Process to Service Dispatcher I/O debugging
20	x100000	1048576	Basic statistics debugging
21	x200000	2097152	Extra statistics debugging
24	x1000000	16777216	Log PORT_ACCESS mapping, DNS_VERIFY_DOMAIN, and ENABLE_RBL denials to the `dispatcher.log` file

11.7 Web-based Monitoring of the Service Dispatcher

The Multithreaded Service Dispatcher maintains statistics for connections made to services controlled by the Dispatcher.

For each connection the Dispatcher receives, it retains information about the connection including connection times, and source and destination IP addresses. The Dispatcher statistics module uses this information to build more comprehensive statistics. For each Worker Process, the Dispatcher maintains statistics on the current, peak (simultaneous), and total number of connections handled by that process. It also tracks the starting and ending time for each process.

On a "higher" level, the Dispatcher also maintains statistics (including the minimum, average, and maximum) on a per-service basis for the number of concurrent connections, the number of connections received per hour, and the durations of the connections received.

These statistics are viewable with a web browser.

To support viewing the statistics, note that the PMDF HTTP server must be configured to serve out the Dispatcher statistics, and access to the PMDF HTTP server must be enabled. If you have used the web-based PMDF-MTA configuration utility or if you have used the command line Dispatcher configuration utility, `pmdf configure dispatcher` (UNIX and VMS), then you will already have a suitable HTTP server configuration. These configuration utilities as of PMDF V6.0 will also have generated an `HTTP_ACCESS` mapping table that allows access only to systems and subnets listed by you as "internal during the configuration". But if you are running with a configuration generated in an older version of PMDF and if you have not already enabled access to the PMDF Service Dispatcher statistics, then you will need to enable such access by adding to the PMDF mapping file an `HTTP_ACCESS` mapping table with entries such as:

```
HTTP_ACCESS

  127.0.0.1|*|*|*|GET|/dispatcher/*      $Y
  *|*|*|*|GET|/dispatcher/*              $N
  *|*|*|*|*|*                            $N
```

The last entry shown is disabling access to all other HTTP server services.

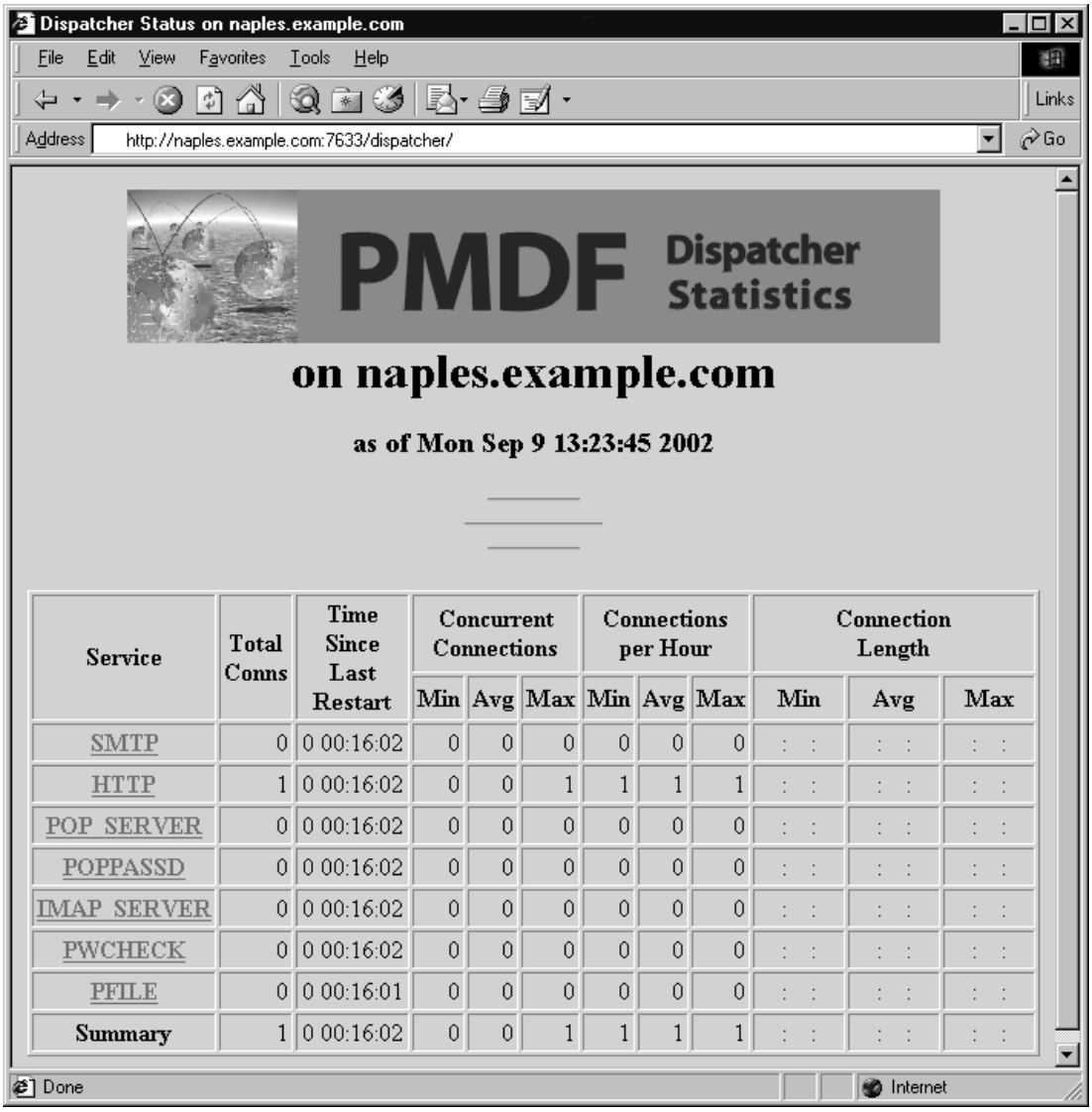
Once the PMDF Service Dispatcher and PMDF HTTP server are configured, access to the Dispatcher statistics has been enabled, and the Dispatcher has been started, you can view the Dispatcher statistics by accessing the URL:

```
http://hostname:7633/dispatcher/
```

where *hostname* is the name of your PMDF system. Figure 11–5 shows an example of Dispatcher statistics display on a sample system.

Note that since the Dispatcher statistics include detailed information regarding what connections were made to your system, it may not be appropriate to make this information publicly available. Just what access permitted is controlled using an `HTTP_ACCESS` mapping table. Another example access mapping is shown here:

Figure 11–5 Dispatcher Statistics Page



Service	Total Conns	Time Since Last Restart	Concurrent Connections			Connections per Hour			Connection Length		
			Min	Avg	Max	Min	Avg	Max	Min	Avg	Max
SMTP	0	0 00:16:02	0	0	0	0	0	0	: :	: :	: :
HTTP	1	0 00:16:02	0	0	1	1	1	1	: :	: :	: :
POP_SERVER	0	0 00:16:02	0	0	0	0	0	0	: :	: :	: :
POPPASSD	0	0 00:16:02	0	0	0	0	0	0	: :	: :	: :
IMAP_SERVER	0	0 00:16:02	0	0	0	0	0	0	: :	: :	: :
PWCHECK	0	0 00:16:02	0	0	0	0	0	0	: :	: :	: :
PFILE	0	0 00:16:01	0	0	0	0	0	0	: :	: :	: :
Summary	1	0 00:16:02	0	0	1	1	1	1	: :	: :	: :

```
HTTP_ACCESS

  *|*|*|*|GET|/dispatcher/              $Y
  127.0.0.1|*|*|*|GET|/dispatcher/*     $Y
  *|*|*|*|GET|/dispatcher/*             $N
```

This will permit anyone to retrieve general information about the Dispatcher statistics, but only the local system (127.0.0.1) will be permitted to get detailed information about individual processes or connections. You can add other IP addresses, such as your workstation from which you run your browser, to this table.

11.8 Tuning System Parameters

The number and type of Dispatcher services offered will affect requirements for various system parameters. The sections below discuss some such operating system parameters.

11.8.1 System Parameters on OpenVMS

Critical SYSGEN parameters for overall Dispatcher functioning

The SYSGEN parameter MAXPROCESSCNT must be large enough to accomodate the number of server processes, in addition to any other processes on the system.

Sites running DEC TCP/IP Services (UCX) should set the SYSGEN parameter

$$\text{CHANNELCNT} = \sum_{services} \text{MAX\_CONNS} + \sum_{services} (\text{MAX\_CONNS} * \text{MAX\_PROCS}) + 30.$$

Sites running MultiNet or TCPware can avoid the need to change their system's CHANNELCNT by instead setting UCX_HOLD=0 in the Dispatcher configuration file.

SYSGEN parameters relevant for mailbox servers

SYSGEN system minimums and system defaults for quotas can affect the quotas for server processes. SYSGEN parameters especially relevant for PMDF mailbox servers are:

PQL_MASTLM, PQL_DASTLM,
PQL_MBIOLM, PQL_DBIOLM,
PQL_MBYTLM, PQL_DBYTLM,
PQL_MCPULM, PQL_DCPULM,
PQL_MDIOLM, PQL_DDIOLM,
PQL_MENQLM, PQL_DENQLM,
PQL_MFILLM, PQL_DFILLM,
PQL_MJTQUOTA, PQL_DJTQUOTA,
PQL_MPGFLQUOTA, PQL_DPGFLQUOTA,
PQL_MTQELM, PQL_DTQELM,
PQL_MWSDEFAULT, PQL_DWSDEFAULT,
PQL_MWSQUOTA, PQL_DWSQUOTA,
PQL_MWSEXTENT, PQL_DWSEXTENT (limited by WSMAX),
VIRTUALPAGECNT, and CHANNELCNT.

Rather than using the SYSGEN values, however, the POP and IMAP servers will preferentially use an explicit quota settings from their Dispatcher configuration file service sections. These SYSGEN values are only used if a Dispatcher quota option has not been set.

The quotas for server processes can be controlled via quota options in the Dispatcher configuration file. The top of the Dispatcher configuration file can define global default quotas, and quota values specific for specific services can be set in that service's definition section in the Dispatcher configuration file.

12 The PMDF HTTP Server

This chapter describes the PMDF HTTP server that serves out PMDF documentation and PMDF monitoring information.

12.1 The PMDF HTTP Server

PMDF comes with a minimal HTTP server that can be used to serve out PMDF documentation. It is also used to support additional management modules, such as Dispatcher statistics. Note that the PMDF HTTP server is not intended for third party use; it is intended purely for serving out PMDF information.

12.1.1 Configuring the HTTP Server

In order to use the HTTP server, the PMDF Service Dispatcher must be configured to handle this HTTP service, the PMDF HTTP server itself must be configured, and access to the HTTP server must be enabled. The PMDF Service Dispatcher command line configuration utility, `pmdf configure dispatcher` (UNIX and VMS), or the PMDF web-based configuration utility, will configure the Dispatcher to handle the HTTP service and will generate a minimal HTTP server configuration file; see the appropriate edition of the *PMDF Installation Guide* for a description and example. *It is highly recommended that the web-based PMDF configuration utility or the Dispatcher command line configuration utility be used to create the HTTP service definition and configuration file.*

Samples of the default HTTP service definition in the Dispatcher configuration file can be seen in Examples Example 12–1 through Example 12–3. A sample of the minimal HTTP server configuration file can be seen in Example 12–4.

Note that the HTTP server listens on port 7633 by default. It can be configured to listen on a different port in the PMDF Service Dispatcher configuration file, `dispatcher.cnf`, located in the PMDF table directory.

The HTTP server configuration file, `http.cnf`, is located in the PMDF table directory. In order for the HTTP server to run properly, this configuration file needs to exist and include entries telling the HTTP server how to handle the various PMDF specific web pages and CGIs, as shown in Example 12–4.

Example 12–1 Sample HTTP Service Definition for OpenVMS

```
!
! HTTP server
!
[SERVICE=HTTP]
PORT=7633
IMAGE=PMDF_EXE:HTTP_SERVER.EXE
LOGFILE=PMDF_LOG:HTTP_SERVER.LOG
```

Example 12–2 Sample HTTP Service Definition for UNIX

```
!
! HTTP server
!
[SERVICE=HTTP]
PORT=7633
IMAGE=/pmdf/bin/http_server
LOGFILE=/pmdf/log/http_server.log
```

Example 12–3 Sample HTTP Service Definition for NT

```
!
! HTTP server
!
[SERVICE=HTTP]
PORT=7633
IMAGE=C:\pmdf\bin\http_server
LOGFILE=C:\pmdf\log\http_server.log
```

The following options can be specified in the HTTP server configuration file:

ALLOW_ROBOTS (integer)
By default, the HTTP server prevents robots from trawling through the HTTP server tree. `ALLOW_ROBOTS` controls the fetching of the file `/robots.txt`. `ALLOW_ROBOTS=0` (the default) will send back a response that should prevent robots from trawling through the HTTP server directory tree, while `ALLOW_ROBOTS=1` will send back a permissive response.

DEBUG (0 or 1)
The `DEBUG` option can be set to 1 to enable debugging of HTTP server activity.

Caution: Since entire transactions are logged, this means that if users use the popstore/MessageStore password changing CGI or set their mailbox filters, then the user's entire transaction *including their password* will be written to the debugging file.

DESCRIPTION (string)
Use the `DESCRIPTION` option to specify an alternate title to appear for the service on the HTTP server main page.

Example 12–4 Sample `http.cnf` File

```
METHODS=GET,POST,HEAD
!
[PATH=/dispatcher/]
GET=PMDF_HTTP_DISPATCHER
!
[PATH=/doc/]
GET=PMDF_HTTP_GET
!
[PATH=/monitor/]
GET=PMDF_MONITOR_CGI
POST=PMDF_MONITOR_CGI
!
[PATH=/qm/]
GET=PMDF_QM_CGI
POST=PMDF_QM_CGI
!
[PATH=/mailbox_filters/]
GET=PMDF_MAILBOX_FILTERS_CGI
POST=PMDF_MAILBOX_FILTERS_CGI
!
[PATH=/configure/]
GET=PMDF_CONFIG_CGI
POST=PMDF_CONFIG_CGI
!
[PATH=/images/]
GET=PMDF_HTTP_GET
HIDDEN=1
!
! The next three lines activate the password change web page
[PATH=/chng_pwd/]
GET=PMDF_POPSTORE_PWD_CGI
POST=PMDF_POPSTORE_PWD_CGI
!
! The next three lines activate the popstore/MessageStore user interface
![path=/msps_user/]
!GET=PMDF_POPSTORE_USER_CGI
!POST=PMDF_POPSTORE_USER_CGI
!
! The next three lines activate the popstore management interface
![path=/popstore/]
!GET=PMDF_POPSTORE_CGI
!POST=PMDF_POPSTORE_CGI
!
! The next three lines activate the MessageStore management interface
![path=/msgstore/]
!GET=PMDF_MSGSTORE_CGI
!POST=PMDF_MSGSTORE_CGI
!
```

DOMAINNAME (string)

Use the `DOMAINNAME` option to specify a hostname that the HTTP server should use in the URLs that it generates. By default, the HTTP server uses the hostname returned by the TCP/IP stack. `DOMAINNAME` is a global option.

GET (string)

The GET HTTP method is used to retrieve documents or information. The value of the GET option should be a shareable image (VMS) or shared library (UNIX) or DLL (NT), which contains the code that implements the GET operation. The value can be a name from the PMDF tailor file.

HEAD (string)

The HEAD HTTP method is similar to the GET method, except that only the header information is retrieved and not the actual contents. The value of the HEAD option should be a shareable image (VMS) or shared library (UNIX) or DLL (NT), which contains the code that implements the HEAD operation. The value can be a name from the PMDF tailor file.

HIDDEN (0 or 1)

By default, all service definitions in the HTTP server configuration file are displayed on the HTTP server main page. Specify the HIDDEN option with a value of 1 on a service definition to prevent that service from being displayed.

LOGGING (0 or 1)

The LOGGING option can be used to cause PMDF to write out a single log line showing successful and failed HTTP requests. Setting LOGGING=1 enables the logging; LOGGING=0, the default, disables it.

Caution: Since entire transactions are logged, this means that if users use the popstore/MessageStore password changing CGI or set their mailbox filters, then the user's entire transaction *including their password* will be written to the log file.

METHODS (string)

The METHODS option is used to declare a list of HTTP methods that the HTML sources can use. The standard ones are GET, POST, and HEAD.

PATH (string)

The PATH option creates a new service definition. Specify the PATH option in square brackets (see Example 12–4). The value of the option should be the subdirectory of the PMDF WWW directory that contains the files for the services. (The PMDF WWW directory is usually PMDF_ROOT:[WWW] on VMS, /pmdf/www on UNIX, and C:\pmdf\wwww on NT.)

PORT (integer)

Specifies the port for the HTTP server to listen on. It only applies to the HTTP server running in standalone mode. For the regular HTTP server, the port is specified in the dispatcher configuration file.

POST (string)

The POST HTTP method is used to submit infromation. The value of the POST option should be a shareable image (VMS) or shared library (UNIX) or DLL (NT), which contains the code that implements the POST operation. The value can be a name from the PMDF tailor file.

REDIRECT (string)

Specifying the REDIRECT option on a service definition causes PMDF to refer over to a different service definition for what pages should be displayed and what actions should be done. The value of the REDIRECT option is the path of another service definition in the HTTP server configuration file.

12.1.2 Access Control

Access to the HTTP server is disabled by default. Access is controlled via the HTTP_ACCESS mapping table. An HTTP_ACCESS mapping table, even one with no entries, allows all access. That is, the line

```
HTTP_ACCESS
```

in your PMDF mappings file is equivalent to

```
HTTP_ACCESS
  *|*|*|*|*|*                          $Y
```

If you want to allow restricted access to services and files offered by the HTTP server, you can use an HTTP_ACCESS mapping table with entries controlling particular sorts of access. If such a mapping table exists in your PMDF mappings file, then on any attempt to connect to the HTTP server port PMDF will probe the table with a probe string of the form

source-ip|*source-port*|*destination-ip*|*destination-port*|*method*|*path*

where *source-ip* is the source IP address (on the remote browser), *source-port* is the source port number (on the remote browser), *destination-ip* is the destination IP address (on the PMDF system), *destination-port* is the destination port number (on the PMDF system), *method* is the HTTP method (about which more below), and *path* is the directory path to the information from the URL. The HTTP method is usually GET, POST, or HEAD: GET is used to retrieve documents or information, HEAD is similar to GET except that only the header information is sent and not the actual contents, POST is often used with forms when submitting information. An asterisk, *, can be used as a wild card in any of the probe string fields.

If the probe string matches a pattern (*i.e.*, the left hand side of an entry in the table), then the resulting output of the mapping is checked. If the output contains any of the metacharacters $Y, $y, $T, or $t, then the access will be permitted. If the mapping output contains any of the metacharacters $N, $n, $F, or $f, then the access will be denied.

For example, to block people from outside the local network (assuming a local network of 192.0.2.*) from viewing PMDF documentation, PMDF monitoring information, or PMDF Service Dispatcher statistics on your server, you could use a mapping table such as:

```
HTTP_ACCESS
  $(192.0.2.0/24)|*|*|*|GET|/doc/*          $Y
  $(192.0.2.0/24)|*|*|*|GET|/monitor/*      $Y
  $(192.0.2.0/24)|*|*|*|GET|/dispatcher/*   $Y
  *|*|*|*|GET|*                             $N
```

Figure 12–1 PMDF HTTP Server Main Page: Services and Documents Available

The default, if no match is found, allows access to any path configured in the HTTP configuration file.

12.1.3 Available Information

Once the HTTP service has been configured, appropriate sorts of HTTP server access have been enabled via a suitable HTTP_ACCESS mapping table, and the PMDF Service Dispatcher has been started up, thereby starting the HTTP server, connect to

```
http://hostname:7633/
```

to see a menu, as in Figure 12–1, of the services and documents available via the PMDF HTTP server.

13 POP and IMAP Mailbox Servers

Mailbox servers allow computers running a client program to access mail residing on the server. POP3 and IMAP4 are protocols for providing such capabilities. PMDF provides POP and IMAP servers. These servers can be used with clients running on PC, Macintosh, UNIX, OpenVMS, or other computer systems to acccess mailboxes on the system running PMDF.[1]

On NT, PMDF provides a POP server and an IMAP server that serve out PMDF MessageStore mailboxes; this POP server can also serve out PMDF popstore mailboxes.

On OpenVMS and UNIX, PMDF currently provides two different POP3 servers and two different IMAP4rev1 servers. The two different POP3 servers provided for these platforms will be referred to as the legacy mailbox POP server and the PMDF MessageStore mailbox POP server; the two different IMAP servers provided will be referred to as the legacy mailbox IMAP server and the PMDF MessageStore mailbox IMAP server. The legacy mailbox IMAP and POP servers serve out VMS MAIL mailboxes (OpenVMS) or BSD mailboxes[2] (UNIX); the legacy mailbox POP servers can also serve out PMDF popstore mailboxes. The PMDF MessageStore mailbox IMAP and POP servers serve out PMDF MessageStore mailboxes; the PMDF MessageStore mailbox POP server can also serve out PMDF popstore mailboxes.

Table 13–1 summarizes the above descriptions of the mailboxes supported by each server.

Table 13–1 IMAP and POP Server Mailbox Support

Platform	Server	Legacy mailbox (VMS MAIL or UNIX BSD)	MessageStore mailbox	popstore mailbox
OpenVMS	legacy POP	yes	no	yes
OpenVMS	MessageStore POP	no	yes	yes
OpenVMS	legacy IMAP	yes	no	no
OpenVMS	MessageStore IMAP	no	yes	no
UNIX	legacy POP	yes	no	yes
UNIX	MessageStore POP	no	yes	yes
UNIX	legacy IMAP	yes	no	no
UNIX	MessageStore IMAP	no	yes	no
NT	MessageStore POP	no	yes	yes
NT	MessageStore IMAP	no	yes	no

[1] Normally the POP and IMAP clients do not run on the PMDF system itself; the PMDF system is the server system serving out mailboxes to POP and IMAP clients on other systems. But see the appropriate edition of the *PMDF User's Guide* for information on Pine, an IMAP or POP client that runs on the PMDF system, which is provided.

[2] Note that the location of those BSD mailboxes can be controlled by PMDF profile database entries.

Note that PMDF fully supports running multiple POP or IMAP servers on non-standard ports; OpenVMS or UNIX sites can run both supplied PMDF POP servers and both supplied PMDF IMAP servers, if they want, by running the different servers on different ports.

PMDF's POP and IMAP servers can be run on any system with a PMDF or PMDF-MTA license.

13.1 POP and IMAP Standards

The current POP3 standard is RFC 1939,[3] and the current IMAP4rev1 standard is RFC 3501.[4] Related RFCs are RFC 1731 (IMAP4 authentication mechanisms), RFC 1733 (Distributed electronic mail models in IMAP4), RFC 2342 (IMAP NAMESPACE command), and RFC 2449 (POP3 CAPA command). PMDF's message store IMAP server also supports RFC 2086 (IMAP4 ACL extension), RFC 2087 (IMAP4 QUOTA extension), RFC 2088 (IMAP4 non-synchronizing literals), and RFC 2359 (IMAP4 UIDPLUS extension)) For user authentication during IMAP or POP (or SMTP) connections, PMDF supports RFC 2222 (SASL; Simple Authentication Security Layer) and RFC 2554 (ESMTP AUTH).

Other RFCs supported in the legacy IMAP server as of PMDF V6.5 include RFC 3516 (IMAP4 BINARY extension), RFC 3348 (IMAP4 CHILDREN extension), RFC 4731 (IMAP4 ESEARCH extension), RFC 2177 (IMAP4 IDLE command), RFC 2088 (IMAP4 LITERAL+), RFC 2193 (IMAP4 MAILBOX-REFERRALS), RFC 4315 (IMAP4 UIDPLUS extension), RFC 3691 (IMAP4 UNSELECT command), and RFC 5032 (IMAP4 WITHIN search extension).

Copies of such recent RFCs can be found in the RFC subdirectory of the PMDF documentation directory, *i.e.*, in `PMDF_DOC:[rfc]` on OpenVMS or in `/pmdf/doc/rfc` on UNIX or in `C:\pmdf\doc\rfc` (possibly on a drive other than `C:`) on NT. Note also that RFCs can be obtained via anonymous FTP from venera.isi.edu.

13.2 Configuring a Mailbox Server

This section describes how to set up a mailbox server. The steps are:

- If you are running any old POP or IMAP server, shut it down; see Section 13.2.1.
- Configure the PMDF Service Dispatcher to handle POP3 or IMAP services; see Section 13.2.2.
- Set POP3 or IMAP server specific configuration options; see Section 13.2.3.
- Start or restart the PMDF Service Dispatcher, thereby starting up the new services; see Section 13.3.

[3] This current standard obsoletes the older POP standards, including RFCs 918, 937, 1081, 1082, 1225, 1460 and 1725.

[4] This current standard obsoletes the older IMAP standards including RFCs 2060, 1064, 1176, 1203, and 1730.

13.2.1 Disabling Old POP or IMAP Servers

If you are already running a POP3 or IMAP server, (including an old PMDF singlethreaded such server), you need to shut it down before starting up a PMDF multithreaded POP3 server or multithreaded IMAP server.

13.2.1.1 Old POP3 or IMAP Servers on OpenVMS

You must shut down any old POP3 or IMAP server you can be running. For instance, MultiNet and TCPware each comes with its own POP3 server which you must disable before starting the PMDF POP3 server.

To disable the MultiNet POP3 server, use the commands:

```
$ MULTINET CONFIGURE/SERVER
SERVER-CONFIG> DISABLE POP3
SERVER-CONFIG> RESTART
SERVER-CONFIG> EXIT
```

If your version of MultiNet also includes an IMAP server, you will need to disable it as well, using the commands:

```
$ MULTINET CONFIGURE/SERVER
SERVER-CONFIG> DISABLE IMAP
SERVER-CONFIG> RESTART
SERVER-CONFIG> EXIT
```

TCPware servers can be disabled using the TCPware menu-driven configuration utility, invoked via the command:

```
$ @TCPWARE:CNFNET
```

13.2.1.2 Old POP3 or IMAP Servers on UNIX

You must shut down any old POP3 or IMAP server you can be running. Check your `/etc/inetd.conf` file; if it has any POP3 or IMAP services defined, that is, any lines such as

```
pop3    stream  tcp     nowait  root    ...
imap    stream  tcp     nowait  root    ...
```

you must remove those lines. After modifying your `inetd.conf` file, you should restart the inetd daemon. For instance, on UNIX this can be performed by issuing a command such as the following:

```
# kill -1 `cat /var/run/inetd.pid`
```

13.2.2 Configuring Mailbox Servers

In order to use PMDF POP3 or IMAP servers, the PMDF Service Dispatcher must be configured to handle the service. The Service Dispatcher configuration file is normally `dispatcher.cnf` located in the PMDF table directory. There are also optional mailbox server specific configuration files, normally `imapd.cnf` and `pop3d.cnf` for the legacy mailbox servers or `imappop.cnf` for the PMDF MessageStore mailbox servers, that are used to control various server options. The PMDF mailbox servers configuration utility should be used to generate the necessary initial POP3 and/or IMAP service definitions for inclusion in the PMDF Service Dispatcher configuration file, and to generate initial POP3 and/or IMAP server configuration files. See the appropriate edition of the *PMDF Installation Guide* for instructions and an example of using this utility.

Once the Dispatcher configuration file has been modified to include the new service definitions, and any desired server configuration options have been set, you must restart the PMDF Service Dispatcher so that it will start the new services, or start it if it was not running previously; see Section 13.3 below for details.

13.2.3 Mailbox Server Configuration Options

There are two (three on OpenVMS) sorts of configuration options relating to the mailbox servers:

- PMDF Service Dispatcher configuration options, controlling things such as the existence of the POP and IMAP servers, and the number of connections handled per server process;
- mailbox server specific options, controlling things such as timeouts, debugging, and logging; and
- (OpenVMS only) the `PMDF_SYSTEM_FLAGS` logical, controlling handling of DECnet format *node*`::`*user* addresses.

These options are discussed in Section 13.2.3.1, Section 13.2.3.2, and Section 13.2.3.3, respectively, below.

13.2.3.1 Service Dispatcher Configuration for Mailbox Servers

In order to run IMAP or POP servers, the Service Dispatcher must be configured to handle such services. Details on the Service Dispatcher can be found in Chapter 11. And the PMDF mailbox servers configuration utility, as discussed in the appropriate edition of the *PMDF Installation Guide*, can and should be used to generate an initial file for inclusion in the Service Dispatcher configuration.

Samples of service definitions for the legacy mailbox servers to be included in the Dispatcher configuration file, `dispatcher.cnf`, are shown in Example 13–1 for OpenVMS, and similarly in Example 13–3 for UNIX. The PMDF mailbox servers configuration utility, if told you want to run only the legacy mailbox servers, would generate similar definitions as the file `dispatcher_mailbox_servers.cnf` in the PMDF table directory. Sample of service definitions for the PMDF MessageStore mailbox

servers to be included in the Dispatcher configuration file are shown in Example 13–2, Example 13–4, and Example 13–5, for OpenVMS, UNIX, and NT, respectively. See Chapter 11 for details on the format and meaning of options in this file.

In particular, on OpenVMS and UNIX where two alternate POP servers and two alternate IMAP servers are available, note that the IMAGE Dispatcher option is used to select the desired server image: compare Example 13–1 *vs.* Example 13–2, or Example 13–3 *vs.* Example 13–4. Also note that sites can run both legacy mailbox servers and PMDF MessageStore mailbox servers on the same system by running them on different ports, as selected by the PORT option.

Example 13–1 Sample `dispatcher_mailbox_servers.cnf` File on OpenVMS for Legacy Mailbox Servers—Dispatcher Definitions for POP and IMAP Servers

```
!
! POP3 server for both POPSTORE and regular mail
!
[SERVICE=POP3]
PORT=110
IMAGE=PMDF_EXE:POP3D.EXE
LOGFILE=PMDF_LOG:POP3D.LOG
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=4
MIN_CONNS=3
MAX_CONNS=5
MAX_LIFE_CONNS=400
PGFLQUOTA=110000
ENQLM=1000
FILLM=450
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
IMAGE=PMDF_EXE:IMAPD.EXE
LOGFILE=PMDF_LOG:IMAPD.LOG
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=4
MIN_CONNS=2
MAX_CONNS=5
MAX_LIFE_CONNS=100
PGFLQUOTA=125000
ENQLM=1000
FILLM=450
```

Example 13–2 Sample `dispatcher_mailbox_servers.cnf` File on OpenVMS for PMDF MessageStore Mailbox Servers—Dispatcher Definitions for POP and IMAP Servers

```
!
! POP3 server for downloading mail
!
[SERVICE=POP_SERVER]
PORT=110
IMAGE=PMDF_EXE:POP_SERVER.EXE
LOGFILE=PMDF_LOG:POP_SERVER_%s.LOG
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=4
MIN_CONNS=3
MAX_CONNS=5
PGFLQUOTA=100500
ENQLM=146
FILLM=187
!
! IMAP4rev1 server for accessing mail on server
!
[SERVICE=IMAP_SERVER]
PORT=143
IMAGE=PMDF_EXE:IMAP_SERVER.EXE
LOGFILE=PMDF_LOG:IMAP_SERVER_%s.LOG
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=4
MIN_CONNS=3
MAX_CONNS=5
PGFLQUOTA=100500
ENQLM=146
FILLM=187
```

13.2.3.2 Mailbox Server Specific Options

IMAP server options are described in Section 13.2.3.2.1 below. POP3 server options are described in Section 13.2.3.2.2 below.

Note that after a server has started running, it will not see changes made to the configuration option files. Use the PMDF RESTART (OpenVMS) or `pmdf restart` command (UNIX or NT) to restart your servers after making changes to these options. Note that if the server is not running, then the Dispatcher itself must be restarted (or started up if not already running) in order to start the server, via the PMDF RESTART DISPATCHER (OpenVMS) or `pmdf restart dispatcher` (UNIX or NT) command, or the PMDF STARTUP DISPATCHER (OpenVMS) or `pmdf startup dispatcher` (UNIX or NT) command.

Example 13–3 Sample `dispatcher_mailbox_servers.cnf` File on UNIX for Legacy Mailbox Servers—Dispatcher Definitions for POP and IMAP Servers

```
!
! POP3 server for both POPSTORE and regular mail
!
[SERVICE=POP3]
PORT=110
IMAGE=PMDF_EXE:pop3d_dispatcher
LOGFILE=PMDF_LOG:pop3d_dispatcher.log
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=3
MIN_CONNS=3
MAX_CONNS=5
USER=root
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
IMAGE=PMDF_EXE:imapd_dispatcher
LOGFILE=PMDF_LOG:imapd_dispatcher.log
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=3
MIN_CONNS=2
MAX_CONNS=5
USER=root
```

13.2.3.2.1 IMAP Server Configuration Options

The legacy mailbox IMAP server has its own configuration file, while the PMDF MessageStore mailbox IMAP server has a separate configuration file shared with the PMDF MessageStore mailbox POP server. The legacy mailbox IMAP server's configuration options are stored in a file located via the `PMDF_IMAP_CONFIG_FILE` logical name (OpenVMS) or PMDF tailor file option (UNIX) or NT Registry entry (NT); by default, the legacy mailbox IMAP server configuration file is `imapd.cnf` located in the PMDF table directory. The PMDF MessageStore mailbox IMAP server's configuration options are stored in a file located via the `PMDF_IMAPPOP_CONFIG_FILE` logical name (OpenVMS) or PMDF tailor file option (UNIX) or NT Registry entry (NT); by default, the PMDF MessageStore mailbox IMAP and POP server configuration file is `imappop.cnf` located in the PMDF table directory. For either IMAP server, the configuration file is optional; if it does not exist reasonable default values will be used.

In either case, the IMAP server option file follows the format of PMDF option files; see, for instance, Section 7.2 for a description of this format.

Note that any changes to an IMAP server configuration file will not take effect until the IMAP server is restarted via the OpenVMS command

```
$ PMDF RESTART/CLUSTER IMAP
```

or the UNIX command

Example 13–4 Sample `dispatcher_mailbox_servers.cnf` File on UNIX for PMDF MessageStore Mailbox Servers—Dispatcher Definitions for POP and IMAP Servers

```
!
! POP3 server for downloading mail
!
[SERVICE=POP_SERVER]
PORT=110
IMAGE=pmdf_exe:pop_server
LOGFILE=pmdf_log:pop_server.log
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=4
MIN_CONNS=3
MAX_CONNS=5
!
! IMAP4rev1 server for accessing mail on server
!
[SERVICE=IMAP_SERVER]
PORT=143
IMAGE=pmdf_exe:imap_server
LOGFILE=pmdf_log:imap_server.log
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=4
MIN_CONNS=3
MAX_CONNS=5
```

```
# pmdf restart imap
```

or the NT command

```
C:\> pmdf restart dispatcher
```

DEBUG (integer)

For both the PMDF legacy mailbox server and the MessageStore mailbox server, this option takes a value where each bit corresponds to a different subcomponent or set of debugging information. This option should only be specified when debugging is needed. Performance of the server is negatively impacted when debugging is enabled.

For the legacy mailbox server, when this option is set to 1 or higher, the POP or IMAP protocol dialogue between the client and server, on a per thread basis, is written to files in the PMDF log directory named `imap_thread.log`.

For the MessageStore mailbox server, if the option is set to 1 or higher, per thread debug logging is written to files in the PMDF log directory named `imaps_thread.log`. The POP or IMAP protocol dialogue output is turned on for the MessageStore servers by setting this option to a value of 7.

If you are having a problem that needs debug logging turned on, Process Software's technical support department will instruct you to set this option's value as needed to

Example 13–5 Sample `dispatcher_mailbox_servers.cnf` File on NT for MessageStore Mailbox Servers—Dispatcher Definitions for POP and IMAP Servers

```
!
! POP3 server for both POPSTORE and MessageStore mail
!
[SERVICE=POP3]
PORT=110
IMAGE=C:\pmdf\bin\pop_server
LOGFILE=C:\pmdf\log\pop_server.log
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=3
MIN_CONNS=3
MAX_CONNS=5
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
IMAGE=C:\pmdf\bin\imap_server
LOGFILE=C:\pmdf\log\imap_server.log
MIN_PROCS=1
MAX_PROCS=6
MAX_SHUTDOWN=3
MIN_CONNS=2
MAX_CONNS=5
```

capture debug logging for the components that are related to the problem.

Be sure to specify this option in the appropriate IMAP server configuration file, `imapd.cnf` for the legacy mailbox server or `imappop.cnf` for the PMDF MessageStore mailbox server, if you want to enable debugging for the IMAP server, not in the Dispatcher configuration file, which enables entirely different debugging! The default is 0.

Caution: Since all input from the client is logged, this means that passwords are also written to the debugging file.

DEBUG_USER_*username* (integer)

For the legacy mailbox server, when this option is set to 1, the POP or IMAP protocol dialogue between the client and server, on a per thread basis, is written to files in the PMDF log directory named `imapd_thread.log`, showing all transactions involving only user *username*.

For the PMDF MessageStore mailbox server, if this option is set to 7, the POP or IMAP protocol dialogue between the client and server, on a per thread basis, is written to files in the PMDF log directory named `imaps_thread.log` showing all transactions involving user *username*. Note that the additional debug logging in the MessageStore mailbox server cannot be turned on by this user-specific debug option, only by the global `DEBUG` option. This option can only turn on the POP/IMAP protocol exchange between the client and server.

The debugging output includes all dialog between the server and the user's client, therefore this should only be used when debugging is needed. Performance of the server will suffer when this is enabled. Be sure to put this in the appropriate IMAP server

configuration file, imapd.cnf for legacy mailbox server or imappop.cnf for the PMDF MessageStore mailbox server, if you want to enable debugging for the IMAP server, not in the Dispatcher configuration file, which enables entirely different debugging! The default is 0.

Caution: Since all input from the client is logged, this means that the user's password is also written to the debugging file.

FORCE_CHECK_TIME (integer; OpenVMS legacy mailbox server only)

This option is an integer value in seconds. On OpenVMS, the server will reopen the NEWMAIL folder and get the current new mail count on a CHECK command if this many seconds have passed since the last CHECK command. This is a way to trade off accuracy versus server overhead because opening a folder is an expensive operation for the VMS MAIL mailbox. The default is 600. A zero value means the server will not automatically reopen the folder.

FORCE_COPY_TO_APPEND (0 or 1; OpenVMS legacy mailbox server only)

On OpenVMS, this option can be used to select OpenVMS mailbox style movement of messages—whereby messages are inserted in original delivery date order into folders when moved—rather than the standard IMAP style movement whereby messages are always inserted at the end of a folder when moved (getting a new date in the process). The default is 1, meaning that standard IMAP style moves are performed. Setting FORCE_COPY_TO_APPEND=0 selects OpenVMS mailbox style movement. Note that using FORCE_COPY_TO_APPEND=0 is not safe unless the IMAP client properly pays attention to UID validity, since the movement of messages into the middle of a folder invalidates UIDs; note that it will also require the client to perform more work, since the client will need to update UIDs after such a message move is performed and UIDs are invalidated.

FORCE_KILL_TIMEOUT (integer)

In the IMAP protocol, a client can connect and stay connected essentially indefinitely, with the client issuing periodic commands keeping the connection alive—typically the frequency of such commands being specified by the IMAP client as an option along the lines of "how often to check for new mail". Unless the user says never to check for new mail, the connection will stay alive indefinitely. Even restarting the PMDF IMAP server won't affect this, because "live" connections must terminate before the IMAP server itself can shut down. This can be undesirable because it means system resources are wasted.

The FORCE_KILL_TIMEOUT option allows the IMAP server to force a disconnect after the specified time if it goes into the shutdown state, (as when its Dispatcher configured MAX_LIFE_CONNS value is reached, or if the PMDF IMAP server or the Dispatcher in general has been instructed to restart). The value for this option specifies the number of seconds to wait before shutting down the connection, when in a shutdown state. For instance, if FORCE_KILL_TIMEOUT=300 is set, then the users connected to the server shutting down will be disconnected after 300 seconds. Warnings are issued to the client, and the client program should display them. The user then has to "Connect" or just open a message, and the client should then reconnect automatically. However, note that the appearance of the warnings ("dialogue boxes") can cause confusion to the users, since the underlying issue (that the server needs to restart) is outside their frame of reference. A different option that can be preferable for limiting the resources used by IMAP users is SESSION_LIFETIME, as its timing depends instead upon a user's own connection lifetime and hence its impact will be more predictable from the user point of view.

The default is that the FORCE_KILL_TIMEOUT option is not set.

See the related options SEND_KILL_WARNING, KILL_WARNING_TEXT, and KILL_FINAL_TEXT.

LOGGING (0, 1, or 2)

When this option is set to 1, user login/logout events will be logged to a PMDF log file. By default, such connection log records are written to the PMDF mail log file `mail.log_current` in the PMDF log directory, but if the `SEPARATE_CONNECTION_LOG PMDF` option has been set, then they will instead be written to the PMDF connection log file, `connection.log_current`. Connection activity is logged with both a time stamp and client host information when each connection is opened, closed, *etc.* When this option is set to 2, only login authentication failures (A records) will be logged to the PMDF log file. The default is 0, meaning that no such logging is performed. The types of entry records written are:

Entry	Description
A	Authentication attempt failed
O	Login phase completed (either successful login or aborted connection)
C	Connection closed cleanly
X	Connection aborted (by either end)†

†Some IMAP/POP clients close the connection without sending a LOGOUT/QUIT command, so an "X" entry can happen in normal operation with such clients.

KILL_FINAL_TEXT (string)

The option is only relevant if `FORCE_KILL_TIMEOUT` is set to a non-zero value. If so, this option controls the text of the final "* BYE" message from the IMAP server before it disconnects clients. The default is "connection shutting down, try reconnect later".

KILL_WARNING_TEXT (string)

The option is only relevant if `FORCE_KILL_TIMEOUT` is set to a non-zero value and if `SEND_KILL_WARNING` is not set to zero. If so, this option controls the text of the warning string sent when the IMAP server wants to shut down. The default is "connection shutting down". To this string, the server will append "in *x* minutes" where *x* is the `FORCE_KILL_TIMEOUT` value.

SEND_KILL_WARNING (0 or 1)

The option is only relevant if FORCE_KILL_TIMEOUT is set to a non-zero value. Setting SEND_KILL_WARNING=0 causes the IMAP server not to send the usual warning due to use of FORCE_KILL_TIMEOUT. The default is 1, meaning to send a warning when FORCE_KILL_TIMEOUT is to be applied.

See also the related option KILL_WARNING_TEXT.

SESSION_LIFETIME (integer)

This option specifies the a length of time, in seconds, after which the IMAP server will terminate an existing IMAP session. That is, when the IMAP server receives a command from the IMAP client, if the client session has existed longer than the specified number of seconds, the IMAP server will terminate that session. This option should be used if a site wants to control the resources used by IMAP users. This option may not be popular with users, but can be useful at sites that do not want to allow individual users to keep IMAP connections constantly open (perhaps blocking use of IMAP by other users). The default is 0, which means there is no limit.

TIMEOUT (integer)
This integer specifies how long (in seconds) the client is allowed to sit idle (*i.e.,* not conduct any transactions with the server) before the server will close the connection. The default is 1800 seconds.

UPDATE_LOGIN_TIME (0 or 1) (OpenVMS legacy mailbox server only)
On OpenVMS, if this option is set to 0, PMDF will not update the SYSUAF last non-interactive login time field upon successful IMAP logins. The default is 1, meaning that PMDF will update the SYSUAF last non-interactive login time field.

13.2.3.2.2 POP3 Server Configuration Options

The legacy mailbox POP server has its own configuration file, while the PMDF MessageStore mailbox POP server has a separate configuration file shared with the PMDF MessageStore mailbox IMAP server. The legacy mailbox POP server's configuration options are stored in a file located via the `PMDF_POP3_CONFIG_FILE` logical name (OpenVMS) or PMDF tailor file option (UNIX) or NT Registry entry (NT); by default, the legacy mailbox IMAP server configuration file is `pop3d.cnf` located in the PMDF table directory. The PMDF MessageStore mailbox POP server's configuration options are stored in a file located via the `PMDF_IMAPPOP_CONFIG_FILE` logical name (OpenVMS) or PMDF tailor file option (UNIX) or NT Registry entry (NT); by default, the PMDF MessageStore mailbox IMAP and POP server configuration file is `imappop.cnf` located in the PMDF table directory. For either POP server, the configuration file is optional; if it does not exist reasonable default values will be used.

In either case, the POP server option file follows the format of PMDF option files; see, for instance, Section 7.2 for a description of this format.

Note that any changes to a POP server configuration file will not take effect until the POP server is restarted via the OpenVMS command

```
$ PMDF RESTART/CLUSTER POP
```

or the UNIX command

```
# pmdf restart pop
```

or the NT command

```
C:\> pmdf restart dispatcher
```

DEBUG (0 or 1)
When this option is set to 1, per thread debugging output will be written to `pop3d_thread.log` files for the legacy mailbox server or `pop3s_thread.log` files for the PMDF MessageStore mailbox server; these files will be located in the PMDF log directory. The debugging output includes all dialog between the server and the client, therefore this should only be used when debugging is needed. Performance of the server will suffer when this is enabled. Be sure to put this in the appropriate POP3 server configuration file, `pop3d.cnf` file for the legacy mailbox server or the `imappop.cnf` file for the PMDF MessageStore server, if you want to enable debugging for the POP3 server, not in the Dispatcher configuration file, which enables entirely different debugging! The default is 0.

DEBUG_USER_*username* (0 or 1)

When this option is set to 1, per thread debugging output will be written to files in the PMDF log directory named `pop3d_thread.log` for the legacy mailbox server or `pop3s_thread.log` for the PMDF MessageStore mailbox server; such files will show each transaction involving user *user*. The debugging output includes all dialog between the server and the user's client, therefore this should only be used when debugging is needed. Performance of the server will suffer when this is enabled. Be sure to put this in the appropriate POP3 server configuration file, `pop3d.cnf` for the legacy mailbox server or `imappop.cnf` for the PMDF MessageStore mailbox server, if you want to enable debugging for the POP3 server, not in the Dispatcher configuration file, which enables entirely different debugging! The default is 0.

DISABLE_UIDL(0 or 1)

When this option is set to 1, the POP3 server will not honor the UIDL command. The default is 0. This option is used for the MessageStore POP server. The legacy POP server uses the option `NO_UIDL`.

FUDGE_SIZE (integer) (OpenVMS only)

On OpenVMS, when this option is set to a non-zero value, the POP3 server will not read the entire message to determine its size in bytes, instead it will multiply the number of records with this number to get an approximate size of the message. Then the value is returned in the `LIST` and `STAT` commands, this makes the POP3 server respond faster when there are a lot of messages in the `NEWMAIL` folder. This option is only used with the VMS MAIL message store. If the message is actually retrieved with the `RETR` command, then the actual message size is used in future responses. The default is 0. Recommended non-zero values range from 80 to 256. 80 is assuming each record is about 80 characters long, 256 is the maximum record size.

LOGGING (0, 1, or 2)

When this option is set to 1, user login/logout events will be logged to a PMDF log file. By default, such connection log records are written to the PMDF mail log file `mail.log_current` in the PMDF log directory, but if the `SEPARATE_CONNECTION_LOG PMDF` option has been set then they will instead be written to the PMDF connection log file, `connection.log_current`. Connection activity is logged with both a time stamp and client host information when each connection is opened, closed, *etc*. When this option is set to 2, only login authentication failures (A records) will be logged to the PMDF log file. The default is 0, meaning that no such logging is performed. The types of etnry records written are:

Entry	Description
A	Authentication attempt failed
O	Login phase completed (either successful login or aborted connection)
C	Connection closed cleanly
X	Connection aborted (by either end)†

†Some IMAP/POP clients close the connection without sending a LOGOUT/QUIT command, so an "X" entry can happen in normal operation with such clients.

MAX_MESSAGES (integer)

The `MAX_MESSAGES` option can be used to limit how many messages are visible to the user. If MAX_MESSAGE=*n* is set, then the server will only ever show the first *n* messages to the client; if the user has more than *n* messages stored, they will only be informed of

those first *n*, and not even be aware that there are more messages. Thus note that users would not like this effect if they are "leaving mail on server".

MIN_LOGIN_INTERVAL (integer)
This option limits how often POP3 logins are allowed. If `MIN_LOGIN_INTERVAL` is set to a positive integer value, then a user can not make another POP3 login to the PMDF MessageStore or to the PMDF popstore (or on OpenVMS, to the VMS MAIL message store) within the number of seconds specified. For VMS MAIL message store POP3 users, as opposed to PMDF MessageStore or PMDF popstore POP3 users, note that there are additional aspects involved with use of this option. (1) For the VMS MAIL message store, this feature depends on the legacy mailbox POP3 server updating the non-interactive login time in the SYSUAF file (which is behavior that is disabled if you set UPDATE_LOGIN_TIME=0 in the legacy mailbox server POP3 server configuration file, `pop3d.cnf`). (2) For the VMS MAIL message store, since the POP3 server is not the only entity which can update the non-interactive login time, in some instances, the user could be denied POP3 login if some other network/batch login event has just occurred. This option does not apply to POP3 users accessing the native mailbox on UNIX.

Note that some POP3 clients can be designed in such a way that repeated logins within a short interval arise in their normal use: for instance, Netscape can login just to check whether a user has new mail, and then login again if the user chooses to get messages. Thus using the `MIN_LOGIN_INTERVAL` option to impose a restriction on how often POP3 users can login can inconvenience users of such clients.

MOVE_READ_MAIL (OpenVMS legacy mailbox server only)
When this option is set to 1, the legacy mailbox POP3 server on OpenVMS will move messages read but not deleted to the MAIL folder. The default is 0, meaning to leave them in the NEWMAIL folder. However, if the option is not explicitly set, the PMDF POP server will also check whether the `MULTINET_POP3_FLAGS` logical exists and is set to 2 and if so, will move messages read but not deleted to the MAIL folder.

NO_UIDL (0 or 1)
When this option is set to 1, the POP3 server will not honor the UIDL command. The default is 0. This option is used for the legacy POP server. The MessageStore POP server uses the option `DISABLE_UIDL`.

OVER_QUOTA_MSG_FILE (string)
This option specifies the name of a file containing customized PMDF popstore over quota warning message text. This is relevant if the PMDF popstore option `QUOTA_WARNING` is set to a non-zero value. See the *PMDF MessageStore & popstore Manager's Guide* for details.

TIMEOUT (integer)
This integer specifies how long (in seconds) the client is allowed to sit idle (*i.e.,* not conduct any transactions with the server) before the server will close the connection. The default is 1800 seconds.

UPDATE_LOGIN_TIME (0 or 1) (OpenVMS legacy mailbox server only)
On OpenVMS, if this option is set to 0, the legacy mailbox POP3 server will not update the SYSUAF last non-interactive login time field upon successful POP logins. The default is 1, meaning that the legacy mailbox POP3 server will update the SYSUAF last non-interactive login time field.

13.2.3.3 The PMDF_SYSTEM_FLAGS Logical and DECnet Style Addresses on OpenVMS

On OpenVMS, the following logical name affects legacy mailbox server behavior.

PMDF_SYSTEM_FLAGS (OpenVMS only)
This option is similar to the MAIL$SYSTEM_FLAGS logical name, but currently only bit 0 is used by the mail servers. If bit 0 is set (*i.e.,* the value of PMDF_SYSTEM_FLAGS is 1), then the node is treated as part of a homogeneous cluster. This is only of impact when a mail message to be read lacks an RFC 822 header. In this case the VMS MAIL From: and To: addresses are converted from the DECnet format *node*::*user* to *user*%*node*@*servernode* if bit 0 is clear, or to *user*@*servernode* if bit 0 is set.

If PMDF_SYSTEM_FLAGS is not set, then the value of MAIL$SYSTEM_FLAGS is used. If MAIL$SYSTEM_FLAGS is not defined either, then the node is not treated as part of a homogeneous cluster, and the address would appear as *user*%*node*@*servernode*.

This logical name should be entered in the SYSTEM logical name table as an EXECUTIVE mode logical.

13.2.4 Registering the Services on UNIX

On UNIX, you can want to edit your /etc/services, (or Yellow Pages, NetInfo, *etc.*, equivalent), to register the POP3 and IMAP services; *e.g.*,

```
pop3  110/tcp
imap  143/tcp
```

13.2.5 Placeholder Message in the BSD Mailbox on UNIX

In order to support required IMAP4 features and desired POP3 features, the legacy mailbox IMAP and POP servers can create a special placeholder message as the first message in the mailbox. By default, the text of this placeholder message is:

```
This message contains information needed by the POP and IMAP servers to
operate correctly.  Please ignore this message.
```

If you want to customize the text of this placeholder message, you can create a file /pmdf/table/ignore-msg.txt containing your desired, site-customized text; the text of the file will be used in place of the default text.

13.3 Starting and Stopping a Mailbox Server

POP3 and IMAP servers are controlled and started by the PMDF Service Dispatcher; as described in Section 13.2.2 above, in order to use POP3 and IMAP servers, the Service Dispatcher must first be configured to handle these services.

13.3.1 Starting a Mailbox Server

To start the POP3 and/or IMAP servers, you must have the Service Dispatcher start the service.

If you are already running the Service Dispatcher, then restart it so that it sees the configuration change with the OpenVMS command

```
$ PMDF RESTART DISPATCHER
```

or the UNIX command

```
# pmdf restart dispatcher
```

or the NT command

```
C:\> pmdf restart dispatcher
```

If you were not previously running the Service Dispatcher, start it with the OpenVMS command

```
$ PMDF STARTUP DISPATCHER
```

or the UNIX command

```
# pmdf startup dispatcher
```

or the NT command

```
C:\> pmdf startup dispatcher
```

On OpenVMS, if you want the IMAP or POP3 servers to start during system startup, then you must execute the PMDF STARTUP DISPATCHER command in your system startup procedure *after* your network software has started and *after* `pmdf_startup.com` has been run.

On UNIX, the PMDF installation procedure ensures that the Service Dispatcher is started automatically during system startup, therefore starting up the POP and IMAP servers, if the Service Dispatcher is configured to handle these services.

13.3.2 Stopping a Mailbox Server

On OpenVMS and UNIX, you can shut down individual servers. To shut down a server, issue the OpenVMS command

```
$ PMDF SHUTDOWN server
```

or the UNIX command

```
# pmdf shutdown server
```

with `pop3` (to shut down both legacy mailbox and PMDF MessageStore mailbox POP servers), `pop_server` (to shut down only the PMDF MessageStore mailbox POP server), `imap` (to shut down both legacty mailbox and PMDF MessageStore mailbox IMAP servers), or `imap_server` (to shut down only the PMDF MessageStore mailbox IMAP server), as appropriate, as the *server* parameter. This will cause such servers on your node to exit after the currently open connections are closed.

On NT, to shut down a server you must shut down the Dispatcher itself, *e.g.*,

```
C:\> pmdf shutdown dispatcher
```

or edit the Dispatcher configuration file to remove the service definition in question and then restart the Dispatcher.

VMS

On OpenVMS, to shutdown servers on certain nodes, use the `/NODE` qualifier. To shutdown servers on the entire cluster, use the `/CLUSTER` qualifier.

Shutting down the Service Dispatcher itself with the OpenVMS command

```
$ PMDF SHUTDOWN DISPATCHER
```

or the UNIX command

```
# pmdf shutdown dispatcher
```

or the NT command

```
C:\> pmdf shutdown dispatcher
```

will also cause the POP3 and IMAP servers to shut down, as well as shutting down any other services handled by the Service Dispatcher; see Chapter 11 for details.

13.3.3 Restarting a Mailbox Server

To cause an already running server to restart, use the OpenVMS command

```
$ PMDF RESTART server
```

or the UNIX command

```
# pmdf restart server
```

Restarting will cause all servers on this node to exit when their currently open connections are closed. New servers will be started by the PMDF Service Dispatcher. *Note that such a command will not restart a server if it has already exited.*

On NT, you must restart the Dispatcher in order to restart its services such as POP3 and IMAP; use the command

```
C:\> pmdf restart dispatcher
```

13.4 Location of User BSD Mailboxes on UNIX

On UNIX, if a user has a PMDF profile database entry specifying that their mail is delivered to a non-default location (or if there is a default profile database entry that applies to that user), then the legacy mailbox POP and IMAP servers will also look at that non-default location for the mailbox to serve out. See Section 17.3.2 for a discussion of the profile database.

13.5 User Login Checks for the VMS MAIL Mailbox (OpenVMS)

On OpenVMS, the following SYSUAF checks are performed by the legacy mailbox servers when a user logs in via a remote client.

Note that these only apply if the user's password is being stored in the VMS SYSUAF file. If the PMDF_TABLE:SECURITY.CNF file is configured such that the authentication source being used is something other than SYSTEM (for example, PASSDB or LDAP), then none of these actions are taken.

However, if the VMS SYSUAF file is the authentication source, the following checks are made:

- The primary password is checked. Currently, the secondary password, if any, is not checked. (The concept of a secondary password is not supported by the POP or IMAP protocols.)
- Account expiration time. Users can not log into mail servers if their account has expired.
- The DISACNT, AUTOLOGIN, PWD_EXPIRED [5] bits in the flags field. The user can not log into the mail server if any of the above bits are set. The DISACNT flag corresponds to the AUTHORIZE utility's DISUSER flag.

[5] `loginout.exe` only sets the PWD_EXPIRED bit if the DISFORCE_PWD_CHANGE flag is set at the time that a user with an expired password logs in. Since the DISFORCE_PWD_CHANGE flag is, by default, not set on accounts, usually the PWD_EXPIRED bit is not set, even if the user's password has expired.

- The CAPTIVE bit is no longer checked thereby allowing access to CAPTIVE and RESTRICTED accounts. You can deny such accounts access by setting one of the above SYSUAF flags, or, if that is not sufficient, use an ACL with the `pop3d.exe`, and `imapd.exe` images and grant the appropriate rightslist identifier to the users in accord with your policy. Any user who does not have EXECUTE access to the image will be denied access.

If LOGGING is set to 1 in the `pop3d.cnf` or `imapd.cnf` file, then login failures are logged in a PMDF log file: the `PMDF_TABLE:mail.log_current` file or the `PMDF_TABLE:connection.log_current` file, depending on the setting of the PMDF option `SEPARATE_CONNECTION_LOG`. A login failure OPCOM message is sent to the SECURITY operator on a VMS 5.*x* system; a NETWORK LOGFAIL audit event is logged on an OpenVMS I64, or OpenVMS 6.1 (VAX) or OpenVMS 6.2 (Alpha) or later system.

If the user fails to log in due to an incorrect password, the number of login failures in the SYSUAF is incremented for the user. Furthermore, if the number of login failures exceeds the SYSGEN parameter LGI_BRK_LIM (default 5) and LGI_BRK_DISUSER is set, then the user account is disabled. A login breakin OPCOM message is sent to the SECURITY operator on a VMS 5.*x* system; a NETWORK BREAKIN audit event (instead of a LOGFAIL event) is logged on an OpenVMS I64, or OpenVMS 6.1 (VAX) or OpenVMS 6.2 (Alpha) or later system after LGI_BRK_LIM is reached.

When a login is successful, the last successful non-interactive login time in the SYSUAF is also updated. A successful NETWORK LOGIN audit event is logged in the system security audit log on an OpenVMS I64, or OpenVMS 6.1 (VAX) or OpenVMS 6.2 (Alpha) or later system.

13.6 Authentication and the Password Database

The PMDF security configuration controls among other things the authentication source used by the PMDF POP and IMAP servers; see the discussion in Chapter 14 on how you can customize this for your site.

Typically, however, users accessing the VMS MAIL message store (OpenVMS) or native Berkeley message store (UNIX) would authenticate against the system password file—except that the system password file on OpenVMS or UNIX can not store certain password forms such as that required for CRAM-MD5 authentication (from IMAP or POP clients) or APOP authentication (from POP clients). Thus in order to perform such authentication from clients, another authentication source must also be in use. That additional authentication source can be the PMDF password database.

When using the PMDF password database as the source of authentication information, note that it can contain several entries, one for each allowed service value. The sort of connection (for instance, whether POP or IMAP) will control which service entry is preferentially checked. Queries by the POP server will first check the user's POP service entry, but if such an entry does not exist will fall through to the the user's DEFAULT service entry. Queries by the IMAP server will first check the user's IMAP service entry, but if such an entry does not exist will fall through to the DEFAULT service entry.

The use of service specific password database entries is not typical; typically, users would each simply have one entry, a DEFAULT service entry, used whenever the PMDF password database is queried. But if users do want to use service specific password database entries, while the above description of service specific probes can sound complicated, the goal is simply to query the "natural" password entry for each case.

So typically, before a POP mail client accessing a native OpenVMS or UNIX message store can use the APOP command to authenticate himself, or before an IMAP or POP mail client accessing a native OpenVMS or UNIX message store can use CRAM-MD5 authentication, the user himself (or the system manager on his behalf) must set the user's password (for the DEFAULT service) in the PMDF password database. See Section 14.7 for additional discussion.

Note that users accessing the PMDF MessageStore or PMDF popstore normally authenticate against a PMDF user profile, which *is* suitable for use for all such forms of authentication. Thus such users normally need not have any PMDF password database entry.

13.7 Mailbox Server Connection Logging

When the LOGGING option is enabled in the IMAP server or POP server configuration file, connection log entries will be generated by that server in the PMDF log file—or if the `SEPARATE_CONNECTION_LOG PMDF` option has been set, see Section 7.3.6, then instead in the PMDF connection log file. Such entries can include detail about SASL errors, details which are not revealed over the wire, such as the distinction between a non-existent user and a bad password.

See Section 31.1.2 for a discussion of the format of such PMDF log file or connection log file entries. In particular, the server entries will be of one of the sorts listed in Table 13–2. Note that every connection gets an "O" entry and either a "C" entry or an "X" entry. Any number of "A" entire entries (including none) can be generated by a single IMAP/POP session.

Table 13–2 IMAP and POP Server Log Entry Codes

Entry	Description
A	Authentication attempt failed
O	Login phase completed (either successful login or aborted connection)
C	Connection closed cleanly
X	Connection aborted (by either end)†

†Some IMAP/POP clients close the connection without sending a LOGOUT/QUIT command, so an "X" entry can happen in normal operation with such clients.

Volume II

The *PMDF System Manager's Guide* is in four volumes. Volume I comprises Chapter 1 through Chapter 13. Volume II comprises Chapter 14 through Chapter 28. Volume III comprises Chapter 29 through Chapter 34.

PMDF software products are marketed directly to end users in North America, and either directly or through distributors in other parts of the world depending upon the location of the end user. Contact Process Software for ordering information, to include referral to an authorized distributor where applicable:

Process Software, LLC
959 Concord Street
Framingham, MA 01701 USA
+1 508 879 6994
+1 508 879 0042 (FAX)
sales@process.com

14 Connection Authentication, SASL, and Password Management

This chapter discusses connection authentication and password source control, including SASL support, the POPPASSD server (supporting the ad-hoc password changing mechanism used by, for instance, Eudora), and the PMDF password database.

PMDF's authentication control facilities include:

- Support for SASL (Simple Authentication and Security Layer—see RFC 2222[1])—a means for controlling the mechanisms by which POP, IMAP or SMTP clients identify themselves to the respective server. PMDF's support for SMTP SASL use complies with RFC 2554 (ESMTP AUTH).
- Support for various authentication sources (password sources), regardless of whether the client supports or uses SASL.
- Support for automatically transitioning users between different authentication sources and mechanisms.
- Support for translating between "external usernames" (what the user types into their client as the username) and "internal usernames" (the name of the underlying account on the PMDF system), as well as support for virtual domains.
- Support for fetching auxiliary properties during authentication.

These facilities are controlled by the PMDF security configuration file, discussed below in Section 14.2, by special entries in the PORT_ACCESS mapping table, discussed below in Section 14.3, and by TCP/IP channel configuration choices (in the case of SASL use over SMTP), discussed below in Section 14.4.

14.1 Background Concepts and Terminology

An *authentication mechanism* is a particular method for a client to prove its identity to a server. APOP, PLAIN, CRAM-MD5, and DIGEST-MD5[2] are examples of authentication mechanisms.

An *authentication verifier* (*e.g.*, password) is stored on the server and contains information used to verify a user's identity. The format of the authentication verifier can restrict which mechanisms can be used. The term authentication verifier is preferred in place of password, since while passwords are the most common instance of authentication verifiers, an authentication verifier could also be something like a certificate in an LDAP

[1] A copy of RFC 2222 can be found in the directory `pmdf_root:[doc.rfc]` (OpenVMS) or `/pmdf/doc/rfc/` (UNIX) or `C:\pmdf\doc\rfc \` (NT).

[2] Mechanism names are as defined by SASL (RFC 2222), which is the IETF (Internet Engineering Task Force—the Internet standards body) specification for adding authentication to protocols such as IMAP and POP. For discussions of particular mechanisms, see for instance RFC 2195 documenting CRAM-MD5, RFC 1939 documenting APOP, and RFC 2617 defining HTTP-digest authentication from which DIGEST-MD5 is derived.

directory or the like; usually, however, one can think "password" wherever one sees "authentication verifier".

An *authentication source* is a file, database, interface to an LDAP directory, *etc.*, accessible to the server wherein are stored authentication verifiers for users. The system password file, PMDF user profile passwords (for PMDF MessageStore or PMDF popstore accounts),[3] and the PMDF password database[4] are examples of authentication sources.

A *security rule set* is a set of rules determining which authentication mechanisms and sources are permitted or used by the server. In PMDF the PORT_ACCESS mapping is used to determine the security rule set to apply to an incoming connection, based on IP addresses and ports.

A *user domain* is an independent set of users known to the server. This is useful, for example, if a server wants to support multiple sets of users possibly with overlapping user names. In PMDF the PORT_ACCESS mapping is used to determine the user domain for each incoming connection, based on IP addresses and ports. Only the PMDF MessageStore authentication source (also used for PMDF popstore) supports multiple user domains; for all other sources, or if no user domain is explicitly specified in the PORT_ACCESS mapping, the default user domain is assumed.

SASL (Simple Authentication and Security Layer)[5] is a way to add different authentication mechanisms to Internet protocols such as POP, IMAP, and SMTP. When the connection is opened, the POP, IMAP, or SMTP client can authenticate itself to the respective server.

14.2 The PMDF Security Configuration file

The PMDF security configuration file controls a number of aspects of authentication of incoming connections by servers such as the PMDF POP, IMAP, or SMTP servers, including what authentication source (password source) a server checks, what authentication mechanism (password verification mechanism) is used to check the authentication verifier (password), when SASL is being used what SASL mechanisms are available, and whether to automatically transition users from one authentication source or mechanism to another.

The security configuration file also controls some aspects of authentication for outgoing connections by clients such as the TCP/IP SMTP channel client, such as specifying usernames and passwords for authenticating to a remote server, and what SASL mechanisms to use.

Currently supported authentication sources include the system password file, the PMDF password database, PMDF user profiles (profiles for PMDF MessageStore and PMDF popstore users), LDAP or X.500 directories, authentication via a remote POP server, and site-supplied routines for password checking. For instance, PMDF can be configured so that when a POP user connects they must issue their system password, or

[3] See the *PMDF popstore & MessageStore Manager's Guide*.

[4] See, for instance, Section 14.7.

[5] For a full description of SASL, see RFC 2222, a copy of which can be found in the RFC subdirectory in the PMDF tree.

must issue their PMDF popstore password, or must issue their PMDF password database password.

Currently supported SASL authentication mechanisms include plaintext, APOP, CRAM-MD5, DIGEST-MD5, and anonymous access. For instance, PMDF can be configured to allow APOP authentication by POP clients, or can be configured to allow only CRAM-MD5 authentication by POP clients.

Different sorts of authentication control can be used for different sorts of connections; for instance, a site might want to use different authentication sources or SASL mechanisms for "internal" *vs.* "external" connections; see Section 14.3 below.

A general overview of the PMDF security configuration file, including specifying for which sorts of connections SASL authentication services are offered, can be found in Section 14.2.2; further details on authentication services such as the list of predefined authentication sources and how to define additional sources can be found in Section 14.2.3; a list of the predefined authentication mechanisms can be found in Section 14.2.4.

14.2.1 Location of the PMDF Security Configuration File

The PMDF security configuration file is located via the `PMDF_SECURITY_CONFIG_FILE` logical (OpenVMS) or PMDF tailor file option (UNIX), or Registry entry (NT) and hence is usually the file `security.cnf` located in the PMDF table directory.

If no security configuration file exists, reasonable defaults are assumed by PMDF. A sample security configuration file corresponding to those internal defaults is shipped with PMDF as the file `security.cnf-sample` in the PMDF table directory.

To override PMDF's internal defaults, or specify additional, site-specific settings, create a `security.cnf` file (more specifically, create the file which is pointed to by `PMDF_SECURITY_CONFIG_FILE`), and then update any compiled PMDF configuration and restart the PMDF Dispatcher as discussed in Section 14.2.9.

14.2.2 Format of the PMDF Security Configuration File

The format of the PMDF security configuration file is similar to that of the PMDF Service Dispatcher configuration file or the PMDF Job Controller configuration file. That is, the PMDF security configuration file generally contains lines of the form

option=value

in accordance with the format of PMDF option files.

In addition to such option settings, the file can contain a line consisting of a section and value enclosed in square brackets of the form

[AUTH_SOURCE=*auth-source-name*]

or

[RULESET=*security-rules-set-name*]

or

[USERNAME_TRANSLATE=*translation-type-name*]

or

[PROP_SOURCE=*auxiliary-property-name*]

or

[CLIENT_AUTH=*auth-name*]

An AUTH_SOURCE section defines an authentication source and sets options for that source. It is not necessary to include an AUTH_SOURCE section for predefined authentication sources, unless a site wants to set special options for that source or unless the source is one such as LDAP which has special required options. If a site wants to provide their own authentication sources, an AUTH_SOURCE section defining that source is required.

A RULESET section sets options applying only to the specified sorts of connections. The value of *security-rules-set-name* is either DEFAULT, or a security rule set selected via the PORT_ACCESS mapping; see Section 14.3. Note that the vertical bar character, |, is not permitted in a *security-rules-set-name*.

If a site, via the PORT_ACCESS mapping, sorts connections into security rule sets other than the default rule set, DEFAULT, then the site's security configuration file should have a RULESET section for each such security rule set, describing the authentication rules to use for connections falling into that security rule set.

A USERNAME_TRANSLATE section defines a username translation function. It is not necessary to include a USERNAME_TRANSLATE section for a predefined username translation function. If a site wants to provide their own username translation function, then a USERNAME_TRANSLATE section defining that function is required. See Section 14.2.5 below.

A PROP_SOURCE section defines an authentication plug-in, referred to as an auxiliary properties module, that can fetch per-user attribute values, "auxiliary properties", during the user authentication process; this tends to be of interest to improve efficiency by getting such attributes directly from the authentication source. See Section 14.2.6 below.

A CLIENT_AUTH section sets options for use by the SMTP client when authenticating to a remote SMTP server. The value of auth-name is either DEFAULT or a name selected via the client_auth channel keyword. If a site, via the client_auth keyword, uses more than one set of client authentication information, then the site's security configuration file should have a CLIENT_AUTH section for each such set.

The following general options can be specified in the PMDF security configuration file.

AUTH_METHOD (1 or 2)

This option controls PMDF's behavior when authenticating usernames and passwords against the list of authentication sources and mechanisms specified in an ENABLE option. The default value is 1, which causes PMDF to check each source/mechanism pair from left to right, stopping as soon as it gets a definitive answer, either *pass* or *fail*. A value of 2 causes PMDF to keep checking authentication sources until it gets a *pass* result, or until all sources are exhausted. Specifically, this option changes PMDF's behavior when getting a *fail* result: the default behavior is to stop, while a value of 2 causes PMDF to keep going.

AUXPROP_ENABLE (comma-separated list of auxiliary-property-modules)

This option, when set in a RULESET section, specifies a list of auxiliary property modules to utilize to set auxiliary properties during authentication. Certain auxiliary properties are available from some authentication sources. Or additional auxiliary properties can be made available by site-supplied auxiliary property modules defined via PROP_SOURCE sections.

BASEDN (distinguished-name)

This option must be used in an [AUTH_SOURCE=LDAP] authentication source definition section to specify the point of the Directory Information Tree to which to bind.

ENABLE (comma-separated list of *source*/*mechanism* pairs)

This option specifies a comma separated list of mechanisms to enable. Each item includes the source name, a slash character, /, and a mechanism name. The source name should either be one of the predefined authentication source names (ANONYMOUS, LDAP, LOGIN, MSGSTORE, PASSDB, POPPROXY, or SYSTEM) or a site-supplied authentication source, as described in Section 14.2.3. The possible mechanism values will vary according to the particular authentication source; for instance, among the predefined authentication sources, the ANONYMOUS source supports only the ANONYMOUS mechanism; the LDAP source supports the PLAIN and CRAM-MD5 mechanisms; the POPPROXY and SYSTEM sources support only the PLAIN mechanism; while the PASSDB and MSGSTORE sources support any of APOP, CRAM-MD5, DIGEST-MD5, or PLAIN. (For further details on these predefined authentication mechanisms, see Section 14.2.4.) Site-supplied authentication sources can have their own list of supported mechanisms. The asterisk character *, can be used to refer to all mechanisms supported by that authentication source. For instance,

```
ENABLE=SYSTEM/PLAIN,MSGSTORE/*
```

If the list of source/mechanism pairs includes more than one source supporting a particular mechanism, then the order of the source/mechanism pairs in the list is significant. When verifying with a particular mechanism, the first source (reading from left to right) that supports that mechanism will be checked first for an entry; if no entry is found, then the next source (reading from left to right) that supports that mechanism will be checked for an entry, *etc*. By default, the verification process is halted as soon as one of the authentication sources provides a definitive answer: PASS or FAIL. This behavior can be modified.

If the AUTH_METHOD option is specified with a value of 2, PMDF will continue checking sources until it gets a PASS result (or until all sources are exhausted). That is, a FAIL result will no longer cause PMDF to stop checking.

FUNCTION (entry-point)
IMAGE (logical-pointing-to-shared-image (OpenVMS) or shared-image-name (UNIX) or dynamic-link-library (NT))

In an AUTH_SOURCE authentication source definition section defining a site supplied authentication source, the IMAGE option specifies the shared image to use and the

FUNCTION option specifies the entry point. These options are mandatory for site defined authentication source definition sections.

In a USERNAME_TRANSLATE section defining a site-supplied translation function to be applied to usernames prior to authentication, the IMAGE option specifies the shared image to use and the FUNCTION option specifies the entry point. The IMAGE option is mandatory for site defined username translation functions. The FUNCTION option can be omitted if the entry-point is called sasl_canonuser_init.

In a PROP_SOURCE section defining a site-supplied authentication auxiliary properties module, the IMAGE option specifies the shared image to use and the FUNCTION option specifies the entry point. The IMAGE option is mandatory for site defined authentication auxiliary property modules. The FUNCTION option can be omitted if the entry-point is called sasl_auxprop_init.

On OpenVMS, the value of the IMAGE option must be a system, executive mode logical name that translates to the name of the shared image; on UNIX, the value of the IMAGE option must be the name of the shared image file; on NT, the value of the IMAGE option must be the name of a dynamic link library (DLL).

LDAP_ATTRIBUTE (attribute-name)

This option can be used in an [AUTH_SOURCE=LDAP] authentication source definition section to specify the name of the attribute to use. The default if this option is not specified is "uid". If the LDAP server that you are using is Active Directory, this option should be specified with a value of "sAMAccountName".

LDAP_CACERTFILE (file-name)

This option can be used in an [AUTH_SOURCE=LDAP] authentication source definition section to specify the name of the file containing the Certificate Authority (CA) certificate that should be used. This option is optional. The default if this option is not specified is to look for the default CA certificate file pmdf_table:ldap-cacert.pem.

LDAP_SEARCHACCT_DN (distinguished-name)
LDAP_SEARCHACCT_PASSWORD (password)

By default, PMDF does an anonymous bind to the LDAP server in order to search it for the username to authenticate. Some LDAP servers, such as Active Directory, do not allow anonymous binds. Use these two options to specify a distinguished name and password to use for that binding process. Only used in an [AUTH_SOURCE=LDAP] authentication source definition section.

LDAP_TLS_MODE (1 or 2)

This option can be used in an [AUTH_SOURCE=LDAP] authentication source definition section to specify whether to use TLS. A value of 1 tells PMDF to try to use TLS, but continue without it if TLS is not available. A value of 2 tells PMDF to require TLS. The default is to not use TLS.

LDAP_VERSION (2 or 3)

This option can be used in an [AUTH_SOURCE=LDAP] authentication source definition section to specify the type of LDAP server in use, v2 or v3, and hence the type of query to perform.

MAIL_DOMAIN (domain-name)

When the LOCALMAIL auxiliary properties module is in effect (explicitly, or implicitly because the PASSDB or SYSTEM authentication source is used), then the auxiliary property SASL_AUX_MAILADDR is normally set to the authenticating username plus the official

local host name (PMDF local channel official host name). This option can be used to specify an alternate domain name to use in this auxiliary property.

MECHANISMS (comma separated list of mechanisms)

This option, when used in a CLIENT_AUTH section, specifies which SASL mechansims to use when an SMTP client is authenticating to a remote SMTP server. The possible mechanisms values are: PLAIN, LOGIN, CRAM-MD5, and DIGEST-MD5. If this option is not specified, all mechanisms are tried.

PASSWORD (string)

This option, when used in a CLIENT_AUTH section, specifies the password to use when an SMTP client is authenticating to a remote SMTP server. This option is required in a CLIENT_AUTH section.

RESTRICT (string)

This option can be used as, for instance, RESTRICT=PLAIN:40 to require a key with 40 significant bits be used for encryption before the PLAIN mechanism is allowed.

SERVER (host-name or ip-number)

This option is used in an [AUTH_SOURCE=LDAP] or [AUTH_SOURCE=POPPROXY] section to specify the LDAP server or POP server, respectively, to which to connect for authentication. The syntax is

SERVER=*server-host-name*:*port*

or

SERVER=*server-host-name*

where the *port* number if omitted will be assumed to be the standard port number (389 for LDAP, or 110 for POP).

For POPPROXY, there can be up to three POP servers specified, separated by commas.

TLS_MODE (0 or 1)

This option can be used in an [AUTH_SOURCE=POPPROXY] authentication source definition section to specify whether to use TLS. A value of 1 tells PMDF to try to use TLS. The default is to not use TLS.

TRANSLATE (translation-type-name)

This option can be used in a RULESET section to specify a function to be appled to usernames before authentication; that is, the username provided by the user attempting to authenticate will be transformed as specified by the function and PMDF will use that transformed username when attempting the authentication. The translation type name must either be one of the predefined translation functions, DEFAULT, ASCII-NOCASE, or IDENTITY, or must specify a translation type name defined in a USERNAME_TRANSLATE section.

USER (username)

This option, when used in an [AUTH_SOURCE=ANONYMOUS] authentication source section, determines the specific username under which anonymous authentication can be performed. When this option is used in a CLIENT_AUTH section, it specifies the remote username to use when an SMTP client is authenticating to a remote SMTP server. This option is required in a CLIENT_AUTH section.

The following additional transition-related options can also be specified in the PMDF security configuration file.

TRANSITION_ADD (comma-separated list of *source*/*mechanism* pairs)
This specifies a list of mechanisms to add when a transition occurs.

TRANSITION_CRITERIA (CLIENT, CHANGE, PLAIN)
This specifies the criteria used to transition users. CHANGE will only transition on password change. CLIENT will transition if the client attempts to use a mechanism for which they don't have a proper entry. Note that in such a case, where the client does not actually have a password entry for the desired mechanism, the server will ask the client to authenticate themselves using a plaintext password (authenticating against the configured authentication source) and the server will then automatically create the desired mechanism entry in that authentication source using the same password value. PLAIN will transition whenever the client uses a plaintext password. The default is TRANSITION_CRITERIA=CHANGE.

TRANSITION_DELETE (comma-separated list of *source*/*mechanism* pairs)
This specifies a list of mechanisms to delete when a transition occurs. Not all authentication sources support this. This option will not take effect unless a TRANSITION_RETAIN_USERS option is present.

TRANSITION_DISABLE (comma-separated list of *source*/*mechanism* pairs)
This specifies a list of mechanisms to disable when a transition occurs. Not all authentication sources support this. This option will not take effect unless a TRANSITION_RETAIN_USERS option is present.

TRANSITION_FROM (comma-separated list of *source*/*mechanism* pairs)
This specifies a list of authentication sources to check when transitioning, in addition to those listed in the ENABLE configuration item. Normally, a user must use one of the source/mechanisms enabled via the ENABLE option in order to connect at all. With the TRANSITION_FROM option, PMDF can be configured to allow one-time-only connection using some other source/mechanisms in order to perform the transition to one of the supported (enabled) source/mechanisms. For instance, when a site is setting up new PMDF MessageStore accounts, one might want to configure IMAP and POP service to require authentication using a user's PMDF user profile password (*i.e.*, PMDF MessageStore password). However, if the first time the user connects they do not yet have a PMDF user profile password set, then allow them to connect using their system password, which will automatically become their initial PMDF MessageStore password. This would correspond to:

```
[RULESET=IMAP-RULES]
ENABLE=MSGSTORE/*
TRANSITION_CRITERIA=CLIENT
TRANSITION_FROM=SYSTEM/PLAIN
TRANSITION_ADD=MSGSTORE/APOP
!
[RULESET=POP-RULES]
ENABLE=MSGSTORE/*
TRANSITION_CRITERIA=CLIENT
TRANSITION_FROM=SYSTEM/PLAIN
TRANSITION_ADD=MSGSTORE/APOP
```

assuming that a PORT_ACCESS mapping is in use that sorts IMAP and POP connections into their own rulesets.

TRANSITION_RETAIN_USERS (comma-separated list of *user* usernames)
This specifies a list of users who are exempt from the TRANSITION_DISABLE and TRANSITION_DELETE options. This option must be set—even if only to an empty value—in order for the TRANSITION_DISABLE and TRANSITION_DELETE options to take effect. That is, TRANSITION_RETAIN_USERS= is fine.

14.2.3 Authentication Sources

An authentication source specifies where (and in some cases how) authentication information is stored. A particular sort of authentication source can support one or more sorts of authentication mechanisms, compatible with the underlying storage of the authentication information; that is, some authentication sources will support only one sort of authentication mechanism, whereas other sources can be able to support additional sorts of authentication mechanisms. For instance, a system password file can only support the PLAIN (plaintext) authentication mechanism. Authentication sources can also support custom configuration options.

Authentication sources are configured via a block of the form

```
[AUTH_SOURCE=auth-source-name]
...
```

in the security configuration file. It is not necessary to include an AUTH_SOURCE block for predefined authentication sources, unless setting special options for that source or unless the source is one such as LDAP which has special required options. An AUTH_SOURCE block must, however, be used when defining a site specific authentication source, as discussed below in Section 14.2.3.2.

14.2.3.1 Predefined Authentication Sources

The following authentication source names are reserved:

ANONYMOUS
This is used for anonymous access. If you want to specify a username for anonymous users, you can set the USER option to the desired user name in the [AUTH_SOURCE=ANONYMOUS] authentication source definition block; *e.g.*,

```
[AUTH_SOURCE=ANONYMOUS]
USER=username
```

LDAP
The LDAP source is used when authentication verifiers are stored in an LDAPv2 or LDAPv3 or X.500 directory accessed via an LDAPv2 or LDAPv3 server. Currently, the LDAP source only supports the PLAIN mechanism (plaintext passwords) and the CRAM-MD5 mechanism. Note that this authentication source requires setting two options to site-specific values, so in order to use it, you must define it in an [AUTH_SOURCE=LDAP] section as illustrated below.

```
[AUTH_SOURCE=LDAP]
SERVER=ldap-server-host-name:port
BASEDN=distinguished-name
```

or

```
[AUTH_SOURCE=LDAP]
SERVER=ldap-server-host-name
BASEDN=distinguished-name
```

If the *port* is omitted from the SERVER option, then the standard LDAP port (port 389) is assumed.

When looking for an authentication verifier in an LDAP directory, PMDF searches by default for a uid attribute matching the username which the user typed. If the LDAP_ATTRIBUTE option is specified, then that attribute is used for searching instead of uid. To do this search, by default an anonymous bind is done. If the LDAP_SEARCHACCT_DN and LDAP_SEARCHACCT_PASSWORD options are specified, then the bind is done using the specified distinguished name and password instead.

Note that if the LDAP server is Active Directory then you should specify the LDAP_ATTRIBUTE option as "sAMAccountName", and you should provide values for both LDAP_SEARCHACCT_DN and LDAP_SEARCHACCT_PASSWORD options. For example:

```
[AUTH_SOURCE=LDAP]
SERVER=ldap-server-host-name
BASEDN=distinguished-name
LDAP_ATTRIBUTE=sAMAccountName
LDAP_SEARCHACCT_DN=distinguished-name
LDAP_SEARCHACCT_PASSWORD=password
```

After doing the search, PMDF then does a bind against the LDAP server with the resulting DN and the user-supplied password. The option LDAP_VERSION controls whether an LDAPv2 or LDAPv3 query is made. The default, if this option is not specified, is LDAP_VERSION=3, causing PMDF to perform a v3 query. If querying an LDAPv2 directory, then LDAP_VERSION=2 must be set; this causes PMDF to perform a v2 query (which is less efficient than a v3 query).

Sites using this source should make sure for performance reasons that the uid attribute (or alternate attribute specified by the LDAP_ATTRIBUTE option) is indexed on the LDAP server. Also note that this source is not currently suitable for high volume use, as in this implementation each authentication opens a separate connection to the LDAP server. High volume sites should instead use the MSGSTORE authentication source and arrange to keep it synchronized with their LDAP server.

PMDF has the ability to access LDAP servers using TLS authentication. Note that sites wanting to use LDAP over TLS must make sure that their LDAP server is set up to do TLS. In order to enable TLS, specify the PMDF_TLS_MODE option as 1 (to try TLS) or 2 (to require TLS). You may need to have the Certificate Authority (CA) certificate to be used by LDAP on your PMDF system. If so, by default the CA certificate should be placed in the file pmdf_table:ldap-cacert.pem. If you wish to use a different file, you may specify it using the LDAP_CACERTFILE option. For example, to use TLS:

```
[AUTH_SOURCE=LDAP]
SERVER=ldap-server-host-name
BASEDN=distinguished-name
LDAP_TLS_MODE=1
LDAP_CACERTFILE=ca-cert-file-name
```

LOGIN

The LOGIN source is used to provide the non-standard LOGIN mechanism. (The LOGIN mechanism is similar to PLAIN and offers no additional functionality, but is nevertheless used by some popular clients.) As implemented in PMDF, the LOGIN authentication source provides the LOGIN mechanism as a shell on top of the PLAIN mechanism from other sources. You must have at least one PLAIN mechanism enabled in order to use the LOGIN authentication source. For instance:

```
ENABLE=SYSTEM/*,LOGIN/*
```

or equivalently:

```
ENABLE=SYSTEM/PLAIN,LOGIN/LOGIN
```

MSGSTORE

This is the set of user authentication profiles used by the PMDF MessageStore and PMDF popstore. This authentication source currently supports the CRAM-MD5, DIGEST-MD5, POP and PLAIN mechanisms. (Note that it always stores the password in a format suitable for use by APOP.) Initial user entries in this authentication source must be generated using PMDF MessageStore or PMDF popstore management utilities; see the *PMDF popstore & MessageStore Manager's Guide*.

PASSDB

Initial user entries in this authentication source must be generated using the pmdf password utility. It currently supports the CRAM-MD5, DIGEST-MD5, APOP, and PLAIN mechanisms.

POPPROXY

This source is used to authenticate against a POP server. When used with automatic transitioning options, this source can be used to migrate passwords from a POP server to a new source, even if the exact storage of the passwords on the POP server is unknown. Such password transitioning is generally done in conjunction with migration of messages from a POP server to a new message store, such as the PMDF MessageStore, though note that such message migration is an entirely separate process from the password migration. See the discussion of the pmdf movein utility in the *PMDF popstore & MessageStore Manager's Guide* for a discussion of message migration.

This source only supports the PLAIN mechanism.

In order to use the POPPROXY source, you must set the SERVER option to tell PMDF the host name of the POP server against which to authenticate, and optionally the port number; if the port number is omitted, then the standard POP port of 110 is assumed. Up to three servers may be specified, separated by commas.

To use TLS on the connection to the POP server, specify the TLS_MODE option. A value of 1 will turn on TLS. Note that in order to use TLS, you must specify a port which is dedicated to TLS (the standard port is 995). POPPROXY does not support issuing an STLS command to the standard POP port.

Some examples:

```
[AUTH_SOURCE=POPPROXY]
SERVER=pop.example.com
```

or

```
[AUTH_SOURCE=POPPROXY]
SERVER=pop.example.com:995, pop2.example.com:995
TLS_MODE=1
```

SYSTEM
This is the system password file, that is, the SYSUAF file on OpenVMS, or usually `/etc/passwd` or `/etc/shadow` on UNIX. This authentication source only supports the PLAIN mechanism. Initial user entries in this authentication source must be generated using system utilities.

On OpenVMS, there is support for the PWDMIX SYSUAF flag, however the following special characters are not allowed in the password:

- whitespace
- open parenthesis (
- close parenthesis)
- open brace {
- percent sign %
- asterisk *
- double quote "
- backslash \

14.2.3.2 Site Specific Authentication Sources

You can define your own password/authentication source by specifying a shared image to call. To add an authentication source called *auth-source-name* where *auth-source-name* can be an arbitrary alphanumeric string *other* than those reserved above, include a block defining the new authentication source (after all global options) of the following form. On OpenVMS:

```
[AUTH_SOURCE=auth-source-name]
IMAGE=logical-pointing-to-shared-image
FUNCTION=function-entry-point
...
```

On UNIX:

```
[AUTH_SOURCE=auth-source-name]
IMAGE=shared-image-name
FUNCTION=function-entry-point
...
```

On NT:

```
[AUTH_SOURCE=auth-source-name]
IMAGE=dll-name
FUNCTION=function-entry-point
...
```

The IMAGE option specifies the shared image to use and the FUNCTION option specifies the entry point. Note that on OpenVMS, the IMAGE value must be a system, executive mode logical name translating to the actual shared image; on UNIX, the IMAGE value must be the actual shared image file name; on NT, the IMAGE value must be the name of a dynamic link library (DLL). These options are mandatory for site defined authentication sources. Additional configuration options specific to that authentication source can also be included.

The PMDF authentication services API can be used to add authentication sources; contact Process Software for details.

14.2.4 Authentication Mechanisms

An authentication mechanism specifies how authentication is performed; that is, how the authenticating password is passed back and forth. Supported authentication mechanisms include:

ANONYMOUS
This permits anonymous access.

APOP
This is a mechanism which can only be used with the POP3 protocol. If set for some other sort of service such as IMAP, it has no effect (is ignored). It supplies the user a challenge and performs a one-way function on the challenge and the user's password. This means that the password is never sent over the wire, but what is sent over the wire can be used to test guesses. It also requires that the password be stored in such a way that if someone gains privileged access to the server and is capable of reverse engineering PMDF's storage mechanism, then they can recover all user passwords.

CRAM-MD5
This is similar to APOP, but is suitable for use with other protocols besides POP3. This is marginally safer than APOP as it permits an authentication verifier storage format such that someone who gains privileged access to the server and is capable of reverse engineering PMDF's storage mechanism only gains the ability to use the CRAM-MD5 mechanism to impersonate any user.

DIGEST-MD5
The DIGEST-MD5 mechanism is based upon the HTTP-digest authentication defined in RFC 2617.

LOGIN
LOGIN is a non-standard mechanism, similar to PLAIN, and offering no additional functionality. But some clients, such as Microsoft Exchange, have nevertheless implemented it. Among the distributed PMDF authentication sources, only the LOGIN source supports the LOGIN mechanism.

PLAIN
This passes the user's plaintext password over the network, where it is susceptible to eavesdropping. Unfortunately, most clients require support for plaintext passwords. This is also the only current mechanism which can be used with system password files. When this mechanism is enabled for POP and IMAP connections, it also enables the plaintext login commands in POP and IMAP.

14.2.5 Username Translation Functions

PMDF supports translating between "external usernames" (what the user types into their client as the username) and "internal usernames" (*e.g.*, the name of a PMDF MessageStore or PMDF popstore account). This can, for instance, be used as part of support for "virtual domains": hosting multiple pseudodomain names on a single system.

A RULESET security ruleset definition section can include an option:

```
TRANSLATE=name
```

where *name* would be defined with a USERNAME_TRANSLATE section of the form:

```
[USERNAME_TRANSLATE=name]
IMAGE=unix-path-or-openvms-logical-or-nt-dll
FUNCTION=entry-point
```

The FUNCTION option can be omitted if the entry-point is called sasl_canonuser_init. Contact Process Software for information on writing site-supplied username translation functions.

The following pre-defined username translations are provided:

DEFAULT
Splits the username at a percent character, %, or at sign character, @, and treats the right-hand side as a user/virtual domain. This is the default behavior. The user/virtual domain is used, for example, when authenticating against popstore accounts to correspond to the popstore *user domain*. This value is not supported when authenticating against system accounts.

ASCII-NOCASE
Same as DEFAULT, but converts ASCII characters on the left-hand side to lower case.

IDENTITY
This passes the username through without any translation.

14.2.6 Auxiliary Properties

PMDF supports fetching "auxiliary properties", that is, per-user attributes, during the user authentication process since under some circumstances this can be the most efficient approach. A primary use for this facility is to associate an e-mail address with an authenticated user during SMTP AUTH processing.

A SASL authentication source can provide values for attributes, or a separate auxiliary properties module can provide values for attributes.

A RULESET security ruleset definition can include:

```
AUXPROP_ENABLE=auxprop-module-name,...
```

or

```
AUXPROP_ENABLE=+auxprop-module-name,...
```

An *auxprop-module-name* would be defined in a PROP_SOURCE section of the form

```
[PROP_SOURCE=auxprop-module-name]
IMAGE=unix-path-or-openvms-logical-or-nt-dll
FUNCTION=entry-point
```

The FUNCTION option can be omitted if the entry point is called sasl_auxprop_init. Contact Process Software for information on the API for writing site-supplied auxiliary property modules; note that the API is currently subject to change.

Normally, the auxiliary properties modules are called in order to fill in any attributes they support which haven't already been filled in automatically by an authentication source. An initial plus character, +, as the first character of the AUXPROP_ENABLE option value causes an auxiliary properties module to override values for properties from a previous auxiliary properties module or authentication source.

The supplied pre-defined auxiliary properties modules are:

MSGSTORE
Determine the e-mail address if a PMDF user profile (PMDF MessageStore or PMDF popstore profile) for the user exists and has a store type of popstore or msgstore.

PASSWD (UNIX only)
Look up attributes via the getpwnam API.

LOCALMAIL
Determine the local e-mail address for a user by looking up the local channel official host name and glueing that onto the user name.

However, if the ruleset being applied has the MAIL_DOMAIN option set to a different domain, then that other domain will instead be used to glue onto the user name. Thus if

```
MAIL_DOMAIN=domain
```

is set, this means that if user chris authenticates using that security ruleset, then his email address is chris@*domain* rather than chris@*local-channel-domain*. As this

is implemented by the LOCALMAIL auxiliary properties module, it doesn't override the behavior of the MSGSTORE auxiliary properties module.

DEFAULT
This is the default, and is equivalent on UNIX to having

```
AUXPROP_ENABLE=MSGSTORE,PASSWD,LOCALMAIL
```

or on OpenVMS or NT systems to having

```
AUXPROP_ENABLE=MSGSTORE,LOCALMAIL
```

set in the ruleset being applied.

Note that an initial plus character, +, has no effect in front of DEFAULT.

Note that these modules are designed to do nothing if the caller didn't ask for the appropriate properties. Currently, the auxiliary property of main interest is an "authenticated" e-mail address; for instance, the PMDF SMTP server will ask for the "SASL_AUX_MAILADDR" property (used if the `authrewrite` keyword is present on the incoming TCP/IP channel). And the MSGSTORE and LOCALMAIL auxiliary properties modules supply such a property.

A PROP_SOURCE section need not be present in the security configuration file for the above pre-defined auxiliary properties module, unless it is desired to modify some portion of the modules usual operation. But site-supplied auxiliary property modules must be established via a PROP_SOURCE definition.

When the MSGSTORE authentication source is used, its auxiliary properties behavior is automatically that of the MSGSTORE auxiliary properties module. When the PASSDB or SYSTEM authentication sources are used, their auxiliary properties behavior is automatically that of the LOCALMAIL auxiliary properties module.

14.2.7 Transitioning Between Authentication Sources

Among other things, the PMDF security configuration can be used to cause users' authentication verifiers (passwords)—for instance, the password used when "logging in" during a POP or IMAP connection, or used for authentication between a SASL-enabled client and a SASL-enabled server—to be migrated from one authentication source to another. This is particularly likely to be relevant when users are being automatically migrated from one message store to another—say from the legacy (native) message store to the PMDF MessageStore or to the PMDF popstore. But it also has other applications: for instance, a SASL-enabled client can tell the server to change the storage of the user's password from one mechanism to another; or a site can choose to migrate users' authentication verifiers from a source on the PMDF system (whether system password file, PMDF password database, or PMDF user profiles for PMDF MessageStore and PMDF popstore users) to an external server, such as a RADIUS server.

Such transitioning is controlled via the various TRANSITION_* PMDF security configuration file options, described individually in Section 14.2.2 above. As transitioning involves additional considerations beyond the usual security configuration file considerations, this section presents an additional brief description of transitioning and the use of the TRANSITION_* options in combination.

The TRANSITION_CRITERIA option specifies if and when to transition users' authentication verifiers. The TRANSITION_ADD and TRANSITION_DELETE options control what storage mechanisms to add and delete when transitioning is performed. TRANSITION_DELETE actually deletes that mechanism's storage of the authentication verifier (password); for instance, if one is transitioning away from the PMDF password database, the PMDF password database entry for that mechanism for the user's authentication verifier is actually removed from the database. The TRANSITION_DISABLE option is less drastic than TRANSITION_DELETE: it marks that password as not usable, but does not actually delete the password. For instance, when the system password file is used, TRANSITION_DISABLE on OpenVMS marks the account as DISUSERed. In other words, TRANSITION_DELETE is not normally reversible, other than by manually reentering the password entry back in, whereas TRANSITION_DISABLE is more easily reversible. The TRANSITION_RETAIN_USERS option specifies particular users, typically users such as root or SYSTEM, who are exempt from the TRANSITION_DISABLE and TRANSITION_DELETE options. This would typically be used when you want to force migration of authentication verifiers for normal users, but not for the special privileged accounts. Finally, the TRANSITION_FROM option specifies a list of additional authentication sources to check when transitioning.

14.2.8 Sample Security Configuration Files

Several sample security configuration files are presented, both basic examples immediately below and more sophisticated examples in the following subsections.

Note: These examples are for the legacy IMAP server.

Example 14–1 shows a security configuration file corresponding to the implicit security configuration used if no security file exists.

Example 14–1 Implicit Default Security Configuration

```
[RULESET=DEFAULT]
ENABLE=MSGSTORE/*,PASSDB/*,SYSTEM/*
```

Example 14–2 shows allowing anonymous IMAP access by anyone to the ftp account. It assumes a PORT_ACCESS mapping sorting IMAP connections into their own IMAPRULES ruleset is in place, along the lines of:

```
PORT_ACCESS
  TCP|*|143|*|*    $YIMAP-RULES
```

Example 14–2 Security Configuration Allowing Anonymous IMAP Access to the `ftp` Account

```
[RULESET=DEFAULT]
ENABLE=MSGSTORE/*,PASSDB/*,SYSTEM/*
!
[AUTH_SOURCE=ANONYMOUS]
USER=ftp
!
[RULESET=IMAP-RULES]
ENABLE=MSGSTORE/CRAM-MD5,MSGSTORE/PLAIN,SYSTEM/PLAIN,ANONYMOUS/*
```

To set controls for any POPPASSD servers (see Section 14.6), one would define a [RULESET=POPPASSD-RULES] section and a PORT_ACCESS mapping assigning POPPASSD connections to the POPPASSD-RULES security rule set; for instance, if the only POPPASSD server listens on port 106, then the PMDF mapping file would need to include something like:

```
PORT_ACCESS
  TCP|*|106|*|*    $YPOPPASSD-RULES
```

Then a security configuration file setting specific controls for POPPASSD connections—namely restricting use of POPPASSD to PMDF MessageStore users, PMDF popstore users or to login users who store their POP password in the PMDF password database (and disabling use of POPPASSD to check the system password file)—could be as shown in Example 14–3.

Example 14–3 Security Configuration with POPPASSD Controls

```
[RULESET=DEFAULT]
ENABLE=MSGSTORE/*,PASSDB/*,SYSTEM/*
!
[RULESET=POPPASSD-RULES]
ENABLE=MSGSTORE/*,PASSDB/*
```

14.2.8.1 Sample Security Configuration Files Using Alternate Authentication Sources

Example 14–4 shows adding a Kerberos V4 shared library.

Example 14–4 Security Configuration Using a Kerberos V4 Shared Library on UNIX

```
[AUTH_SOURCE=KERBEROS]
IMAGE=/usr/local/lib/krb4sasl.so
FUNCTION=krb4sasl_init
SRVTAB=/etc/srvtab
!
[RULESET=DEFAULT]
ENABLE=KERBEROS/*,MSGSTORE/*,PASSDB/*,SYSTEM/*
```

Example 14–5 shows a security configuration file for looking up authentication verifiers in an LDAP directory. See the additional discussion of [AUTH_SOURCE=LDAP] in Section 14.2.3.

Example 14–5 Security Configuration for LDAP Authentication

```
[RULESET=DEFAULT]
ENABLE=LDAP/*
!
[AUTH_SOURCE=LDAP]
SERVER=elvira.example.com
BASEDN=o="Example Software",st=Massachusetts,c=us
```

If the LDAP directory does not support CRAM-MD5, or if a site is using {CRYPT} passwords on the LDAP server, then the mechanisms offered should be restricted to PLAIN, as shown in Example 14–6.

Example 14–6 Security Configuration for LDAP Authentication without CRAM-MD5

```
[RULESET=DEFAULT]
ENABLE=LDAP/PLAIN
!
[AUTH_SOURCE=LDAP]
SERVER=elvira.example.com
BASEDN=o="Example Software",st=Massachusetts,c=us
```

14.2.8.2 Sample Security Configuration Files for Transitioning Between Authentication Sources

The examples in this section assume that a PORT_ACCESS mapping sorting connections into their own IMAP and POP rulesets is in place, along the lines of:

```
PORT_ACCESS

  TCP|*|110|*|*    $YPOP
  TCP|*|143|*|*    $YIMAP
```

Example 14–7 shows moving POP users from the system password file to PMDF user profile passwords (PMDF MessageStore and PMDF popstore profile passwords); hence this is the sort of security configuration a site might use when POP users are being transitioned from use of the legacy mailbox (*i.e.*, BSD mailbox on UNIX or VMS MAIL mailbox on OpenVMS) to use of the PMDF popstore mailbox. Example 14–8 shows disallowing use of plaintext passwords; only one time use of plaintext password is allowed as the passwords are migrated to CRAM-MD5 storage. Example 14–9 similarly shows disallowing use of either plaintext or APOP, other than as a one time transitional usage, as passwords are migrated to CRAM-MD5 storage.

Example 14–7 Security Configuration when Migrating POP Users to the PMDF popstore

```
[RULESET=DEFAULT]
ENABLE=PASSDB/CRAM-MD5,PASSDB/PLAIN,SYSTEM/PLAIN
TRANSITION_CRITERIA=CLIENT
!
[RULESET=IMAP]
ENABLE=MSGSTORE/CRAM-MD5,MSGSTORE/PLAIN,PASSDB/CRAM-MD5,PASSDB/PLAIN,SYSTEM/PLAIN
!
[RULESET=POP]
ENABLE=MSGSTORE/*,SYSTEM/*
TRANSITION_CRITERIA=PLAIN
TRANSITION_ADD=MSGSTORE/PLAIN
TRANSITION_DISABLE=SYSTEM/PLAIN
TRANSITION_FROM=SYSTEM
TRANSITION_RETAIN_USERS=admin1,admin2
```

Example 14–8 Security Configuration Disallowing plaintext Passwords, except for Transitioning to CRAM-MD5

```
[RULESET=DEFAULT]
ENABLE=PASSDB/CRAM-MD5
TRANSITION_CRITERIA=CLIENT
TRANSITION_FROM=PASSDB/*,SYSTEM/*
TRANSITION_ADD=PASSDB/CRAM-MD5
!
[RULESET=POP]
ENABLE=MSGSTORE/CRAM-MD5,MSGSTORE/APOP,PASSDB/CRAM-MD5,PASSDB/APOP
!
[RULESET=IMAP]
ENABLE=MSGSTORE/CRAM-MD5,PASSDB/CRAM-MD5
```

Example 14–9 Security Configuration Disallowing plaintext and APOP

```
[RULESET=DEFAULT]
ENABLE=PASSDB/CRAM-MD5
TRANSITION_CRITERIA=CLIENT
TRANSITION_ADD=PASSDB/CRAM-MD5
TRANSITION_DELETE=PASSDB/PLAIN
TRANSITION_FROM=PASSDB/*,SYSTEM/*,MSGSTORE/*
!
! Disable use of the APOP mechanism for the PMDF password database
!
[AUTH_SOURCE=PASSDB]
PASS_FORMAT=CRAM-MD5
!
[RULESET=POP]
ENABLE=MSGSTORE/CRAM-MD5,PASSDB/CRAM-MD5
TRANSITION_FROM=MSGSTORE/*,PASSDB/*,SYSTEM/*
!
[RULESET=IMAP]
ENABLE=MSGSTORE/CRAM-MD5,PASSDB/CRAM-MD5
```

14.2.9 Updates to the Security Configuration

The security configuration file is part of a compiled PMDF configuration. If you are using a compiled PMDF configuration, you will need to recompile and reinstall it after making changes to the security configuration file.

After changes to the security configuration file, (and recompiling, if using a compiled configuration), the PMDF Dispatcher should be restarted with the `pmdf restart dispatcher` command.

14.3 The PORT_ACCESS Mapping: Security Rule Sets and User Domains

The PORT_ACCESS mapping can be used to cause PMDF to classify incoming connections into different *security rule sets* and into different *user domains*.

Security rule sets provide a way of having connection based security differentiation. For instance, a site might want to use different security mechanisms for connections from "internal" *vs*. "external" sources.

A site using the PMDF popstore, (which supports multiple user domains —multiple sets of users with possibly overlapping user names), might also want to authenticate (independent of authentication mechanism) against different sets of user profiles, for instance, "student" profiles *vs.* "faculty" profiles, depending upon the incoming connection; such sets of user profiles are known as *user domains*.

For background information on the PMDF mapping file, see Chapter 5; for an introduction to the PORT_ACCESS mapping in particular, see Section 11.5.

The format of a PORT_ACCESS entry specifying a security rule set is:

```
PORT_ACCESS
  ...     $Y security-rules-set-name
```

and the format of a PORT_ACCESS entry specifying both a security rule set and a user domain is:

```
PORT_ACCESS
  ...     $Y security-rules-set-name|user-domain
```

For instance, Example 14–10 shows a sample PORT_ACCESS mapping that assigns incoming connections according to server port number to security rule sets named POP-RULES, and IMAP-RULES. Example 14–11 shows a sample PORT_ACCESS mapping that assigns incoming connections from IP addresses in the 192.160.253.* subnet to an INTERNAL security rule set, while assigning all other incoming connections to an EXTERNAL security rule set.

Example 14–12 shows a sample PORT_ACCESS mapping that sorts incoming connections into two user domains, VIP and LABRAT, and into INTERNAL and EXTERNAL security rule sets. This server is assumed to have two IP addresses (via multi-homing or two interface cards), 192.160.253.60 and 192.160.253.61, and selects the user domain based on that. The PORT_ACCESS mapping entries shown specify that users in the VIP user domain are allowed to connect to any port (send or read mail) from external systems, whereas users in the LABRAT user domain, while they can connect to the SMTP port to send mail from external systems, are not allowed to connect to POP or IMAP servers to read mail from external systems.

Example 14–10 PORT_ACCESS Mapping for Security Rule Set Based on Server Port Number

```
PORT_ACCESS

  TCP|*|110|*|*    $YPOP-RULES
  TCP|*|143|*|*    $YIMAP-RULES
```

Example 14–11 PORT_ACCESS Mapping for Security Rule Set Based on Source IP Address

```
PORT_ACCESS

  TCP|*|*|$(192.160.253.0/24)|*    $YINTERNAL
  TCP|*|*|*|*                      $YEXTERNAL
```

Example 14–12 PORT_ACCESS Mapping for Distinguishing User Groups

```
PORT_ACCESS

  TCP|192.160.253.60|*|$(192.160.253.0/24)|*  $YINTERNAL|VIP
  TCP|192.160.253.60|*|*|*                    $YEXTERNAL|VIP
  TCP|192.160.253.61|*|$(192.160.253.0/24)|*  $YINTERNAL|LABRAT
  TCP|192.160.253.61|25|*|*                   $YEXTERNAL|LABRAT
  TCP|*|*|*|*                                 $N
```

Note that the PORT_ACCESS mapping table, being part of the PMDF mapping file, is part of a compiled PMDF configuration. If you are using a compiled PMDF configuration, you will need to recompile and reinstall it after making changes to the PORT_ACCESS mapping table. Also, after changes to the PORT_ACCESS mapping table the PMDF Dispatcher should be restarted with the `pmdf restart dispatcher` command.

14.4 SASL Configuration for TCP/IP Channels

Submission of SMTP messages is normally unauthenticated—the SMTP client performs no authentication of who it "really" is and merely submits a message. (See RFC 821.) SASL and specifically the AUTH SMTP extension (see RFC 2222) provides a protocol by which an SMTP client can authenticate itself to the server.

14.4.1 SMTP Server

SMTP server support for SASL can be controlled by various channel keywords, as described in Section 2.3.4.43 and Section 2.3.4.46. The default is that the SMTP server does not advertise nor support SASL use, `nosaslserver`.

Note that the authentication source and mechanisms supported for SASL use by the SMTP server are controlled by the PMDF security configuration file, as discussed above in Section 14.2.

One use of SASL in the SMTP server is to allow authenticated clients to perform message submissions that would be disallowed to unauthenticated clients. For instance, a site that generally blocks SMTP relaying through their SMTP server, but wants to allow such SMTP relaying for specific users who will authenticate themselves using SASL, might use channel definitions along the lines of:

```
tcp_local smtp mx single_sys maysaslserver saslswitchchannel tcp_auth
TCP-DAEMON

tcp_auth smtp mx single_sys mustsaslserver
TCP-AUTH
```

with an `ORIG_SEND_ACCESS` mapping table along the lines of:

```
ORIG_SEND_ACCESS

  tcp_local|*|tcp_local|*     $NSMTP$ relaying$ not$ permitted
```

Here the `tcp_local` channel is assumed to be the "external" TCP/IP channel. An attempt to submit without authentication a message that would go straight back out the `tcp_local` channel will be rejected due to the `ORIG_SEND_ACCESS` entry shown. But if a connection from an external system performs SASL authentication, the connection is switched to the `tcp_auth` channel. The `tcp_auth` channel will not allow messages submission unless the remote connecting client successfully authenticates itself. For connections that do authenticate, the messages will be accepted on the `tcp_auth` channel, and can be relayed out via the `tcp_local` channel, should that be the appropriate destination channel.

A similar example would be for a site that also allows relaying by "internal" clients or systems, using `switchchannel` and rewrite rules to associate and switch "internal" connections – connections from .example.com subdomains or IP addresses in the 1.2.3.0 subnet – to their `tcp_internal` channel. Such a site might use rewrite rules:

```
.example.com      $U%$H$D@TCP-INTERNAL
[1.2.3.]          $U%[1.2.3.$L]@TCP-INTERNAL$E$R
```

and channel definitions along the lines of

```
tcp_local smtp mx single_sys maysaslserver saslswitchchannel tcp_auth \
 switchchannel
TCP-DAEMON

tcp_internal smtp mx single_sys maysaslserver allowswitchchannel
TCP-INTERNAL

tcp_auth smtp mx single_sys mustsaslserver noswitchchannel
TCP-AUTH
```

with an ORIG_SEND_ACCESS mapping table along the lines of:

```
ORIG_SEND_ACCESS

  tcp_local|*|tcp_local|*     $NSMTP$ relaying$ not$ permitted
```

Connections from "internal" systems will be switched to the tcp_internal channel. That channel will permit SASL use (though clients need not bother to use SASL). Connections from external systems that use SASL to authenticate will be switched to tcp_auth. Since the tcp_internal and tcp_auth channels can send out via tcp_local (are not blocked by ORIG_SEND_ACCESS), then messages from internal users or from external users who use SASL authentication will be permitted to be submitted to the Internet. But all other attempted messages submissions from external systems, to attempted Internet destinations, will be rejected due to the ORIG_SEND_ACCESS entry.

TCP/IP channels can also be configured to place the SASL authenticated address in the headers; see Section 2.3.4.44 and Section 16.1.3.

Note that if you are using a compiled PMDF configuration, you will need to recompile and reinstall it after making changes to TCP/IP channels in the PMDF configuration file. Also, after changes to TCP/IP channel definitions, the PMDF SMTP server should be restarted with the command pmdf restart smtp (UNIX or OpenVMS) or pmdf restart dispatcher (Windows).

14.4.2 SMTP Client

PMDF has the ability to configure the TCP/IP channel client to use SASL via the SMTP AUTH command when sending mail out from the PMDF MTA to a remote MTA. This is primarily needed by home users who are running PMDF on their home systems and have an ISP that requires a username and password to be able to send out mail through the ISP's MTA.

The username and password to use for authentication is configured in a CLIENT_AUTH section, as discussed in Section 14.2. An example CLIENT_AUTH section for remote system 'alpha' is as follows:

```
[CLIENT_AUTH=alpha]
USER=alpha-username
PASSWORD=alpha-password
```

The TCP/IP channel also needs to be configured to enable client-side SASL. This is done with one of the following channel keywords: maysaslclient, mustsaslclient, maysasl, or mustsasl. For details see Section 2.3.4.43.

By default, the [CLIENT_AUTH=default] section is used to get the username and password. To use a different CLIENT_AUTH section, specify its name using the client_auth channel keyword.

This example channel definition is used to send mail out to a system called 'alpha' on the SMTP submission port (587) using SASL and TLS.

```
tcp_alpha smtp mx port 587 daemon router maysaslclient allowswitchchannel \
        maytls client_auth alpha
alpha.example.edu
TCP-ALPHA
```

14.5 Recording of SASL Use in Received: Headers and PMDF Log Entries

When an SMTP message is received with SASL, the Received: header PMDF constructs will include the words "with ESMTPA" (if received with SASL only) or "with ESMTPSA" (if received with TLS and SASL) rather than the usual "with SMTP" (received without extended SMTP), "with ESMTP" (received with extended SMTP), or "with ESMTPS" (received with TLS but not SASL).

If the logging channel keyword is enabled, then messages received or sent with SASL used will show an "A" (authentication) character in addition to the usual "E" or "D" character in the mail.log* file entries. If the LOG_USERNAME PMDF option is also set, see Section 7.3.6, then the username field of the mail.log* entries will show the username that authenticated, prefixed with the asterisk, *, character.

14.6 The POPPASSD Server

The POPPASSD server is used to support changing authentication verifiers (passwords) from POP clients, using the ad-hoc password changing mechanism used by, for instance, Eudora. Note that the POPPASSD protocol involves sending both old and new password "in the clear"; sites should consider this when deciding whether they want to provide this service.

The source of the authentication verifier to be changed—whether the system password file, PMDF user profile password (PMDF MessageStore or PMDF popstore password), or PMDF password database, or some site defined source—can be controlled via the PMDF security configuration; see Section 14.2. For instance, with the implicit security rules used by PMDF if not explicit security configuration file exists, the POPPASSD server will attempt to change the password stored in a user's PMDF user profile, PMDF password database, and the system password file. The POPPASSD server will change each occurrence of the user's password, if the password is stored in more than one location (for instance, stored in both the PMDF password database and the system password file). The POPPASSD server will modify only password entries only for those users with existing entries; it will not create a new entry for a user who did not previously have an entry.

When changing a user's password entry in the PMDF password database, the POPPASSD server will preferentially change the user's SERVICE=POP entry (if one exists); if no POP-service-specific entry is present, then the POPPASSD server will instead change the user's SERVICE=DEFAULT entry.

14.6.1 Configuring the POPPASSD Server

The PMDF mailbox servers configuration utility will ask if you want to run a POPPASSD server; you should use that utility to generate the appropriate PMDF Service Dispatcher definition; see the appropriate edition of the *PMDF Installation Guide* for details on using the configuration utility. Samples of the sort of service definition that would be created by that utility are shown in Example 14–13 and Example 14–14.

Example 14–13 Sample POPPASSD Service Definition for the Dispatcher on OpenVMS

```
!
! POP3 password daemon for Eudora
!
[SERVICE=POPPASSD]
PORT=106
IMAGE=PMDF_EXE:POPPASSD.EXE
LOGFILE=PMDF_LOG:POPPASSD.LOG
MIN_PROCS=1
MAX_PROCS=2
MIN_CONNS=2
MAX_CONNS=5
```

Example 14–14 Sample POPPASSD Service Definition for the Dispatcher on UNIX

```
!
! POP3 password daemon for Eudora
!
[SERVICE=POPPASSD]
PORT=106
IMAGE=/pmdf/bin/poppassd
LOGFILE=/pmdf/log/poppassd.log
MIN_PROCS=1
MAX_PROCS=2
MIN_CONNS=2
MAX_CONNS=5
USER=root
```

Once such a service definition has been added to the Dispatcher configuration file, you must restart the PMDF Service Dispatcher so that it will start the new service, or start the Dispatcher if it was not running previously; see Chapter 11 for details.

14.7 The PMDF Password Database

The PMDF password database stores, as the name suggests, passwords. Note that APOP and CRAM-MD5 passwords cannot be stored in the system password file; such passwords must be stored in a particular format which the system password file does not support. Therefore, in order to support use of the POP protocol's APOP command or AUTH command with CRAM-MD5, or the IMAP protocol's AUTHENTICATE command with CRAM-MD5, the user must have a password entry stored in an authentication source other than (or in addition to) the system password file. The PMDF password database can be that additional authentication source.

Note that in general, whether the PMDF password database is consulted at all for authentication is controlled by the PMDF security configuration, as described in Section 14.2. That is, a connection comes in (POP, IMAP, mailbox filtering, or, if SMTP SASL use is enabled, SMTP) and is mapped to a security rule set; the security rule set in the PMDF security configuration then controls where and how authentication is performed for that connection.

For instance, the DEFAULT security rule set in PMDF's implicit security configuration (which applies if no security configuration file exists) checks first for a PMDF user profile password (PMDF MessageStore or PMDF popstore password), next for a PMDF password database entry, and finally falls through to checking for a system password entry.

Thus for instance, for a POP or IMAP connection handled by the DEFAULT security rule set, if a user attempts to authenticate using the APOP or CRAM-MD5 mechanism, that user must either be a PMDF MessageStore or PMDF popstore user (in which case their PMDF MessageStore or PMDF popstore password is normally[1] sufficient for remote authentication), or if they are a legacy message store (VMS MAIL on OpenVMS, or Berkeley mailbox on UNIX) user then they must have a PMDF password database entry in addition to their system password file entry.

For mailbox filter connections handled by the DEFAULT security rule set of PMDF's implicit security configuration, authentication will be performed preferentially against the PMDF user profile (PMDF MessageStore or PMDF popstore user profile), if the user has a user profile entry, if not then against the PMDF password database, if the user has an entry in it, and finally, only if the user has neither sort of entry, against the system password file.

Consider a typical configuration in which SMTP connections are handled by the DEFAULT security rule set of PMDF's implicit security configuration. In this case for an SMTP connection that attempts to authenticate with the ESMTP AUTH command, if CRAM-MD5 authentication is attempted then the user must have a PMDF user profile entry or a PMDF password database entry. If PLAIN or LOGIN authentication is attempted, then the password is checked first against the user's PMDF user profile entry, if one exists, next against the PMDF password database, and finally, only if the user has neither sort of entry, against the system password file.

[1] The PMDF MessageStore and PMDF popstore, however, each have a PWD_ELSEWHERE flag to say that its passwords are stored elsewhere; if this is set, even a PMDF MessageStore or PMDF popstore user might use a PMDF password database entry.

14.7.1 Location of the PMDF Password Database

The PMDF password database is pointed to by the `PMDF_PASSWORD_DATABASE` logical name (OpenVMS) or PMDF tailor file option (UNIX) or Registry entry (NT), and hence is usually the file `password.auth` in the PMDF table directory.

14.7.2 Entries in the PMDF Password Database

The PMDF password database is normally created and modified using the `pmdf password` utility. With this utility the PMDF postmaster can set entries for users. Or users can set and change their own passwords.

Section 14.7 above discusses whether and when the PMDF password database will actually be used as the source of authentication information. When the PMDF password database *is* used as the source of authentication information, then an additional issue can arise, namely which of a user's possibly multiple entries will be checked for the authentication. That is, a user can have multiple entries in the PMDF password database, one for each allowed service value. The sort of connection (assuming that the PMDF password database is even checked) will control which service entry is preferentially checked. Note that the sort of service entry checked has nothing to do with the PMDF security configuration (which instead controlled whether or not the PMDF password database was queried at all); the sort of service entry checked when the PMDF password database is queried has entirely to do with which component of PMDF is doing the querying (what sort of connection this regards).

Queries by the POP server will first check a user's POP service entry, but if such an entry does not exist will fall through to the user's DEFAULT service entry. Queries by the IMAP server will first check a user's IMAP service entry, but if such an entry does not exist will fall through to the user's DEFAULT service entry.

Queries for mailbox filtering will check which channel a user matches. For a user matching the msgstore channel, the mailbox filter query will preferentially use the user's service=IMAP entry, but if such an entry does not exist will fall through to the user's service=DEFAULT entry. For a user matching the popstore channel, the mailbox filter query will preferentially use the user's POP service entry, but if such an entry does not exist will fall through to the user's DEFAULT service entry. For a user matching the local channel, the mailbox filter query will use the user's DEFAULT service entry.

Most sites and users will not want to use service specific password database entries. Then each user has one entry, their DEFAULT service entry, used whenever the PMDF password database is queried.

But for sites and users who do want to use service specific password database entries, while the above description of service specific probes can sound complicated, the goal is simply to query the "natural" password entry for each case.

15 PMDF-TLS: Transport Layer Security

Note: PMDF-TLS is a separately licensed layered product built on top of PMDF.

PMDF-TLS implements the TLS protocol (Transport Layer Security; see RFC 2246) for PMDF's servers and clients. Transport Layer Security is currently supported for:

- SMTP (server and client),
- IMAP,
- POP3, and
- HTTP.

PMDF-TLS provides for a secure data stream between the client and the server so that users can ensure that the data that is exchanged between their system and a remote system using TLS will be protected from others on the network.

Note that TLS is backwards compatible with SSL (Secure Sockets Layer) and PMDF-TLS is fully compatible with SSL-enabled clients. Because TLS includes all necessary SSL functionality, this document will refer to TLS exclusively.

15.1 Overview of Operation

There are two modes of operation that PMDF-TLS supports:

1. Connecting to a TLS-enabled port where TLS negotiation happens immediately once the TCP connection has been established; and
2. Connecting to a "regular" port and then issuing a STARTTLS command[1] to begin TLS negotiation.

The only difference between these two modes is when the TLS negotiation happens. In both cases, once the TLS negotiation is complete, all subsequent data sent across the TCP connection will be secure.

Connecting to a special port number is currently the more commonly used way to connect to a TLS-enabled server, but connecting to a regular port and issuing a STARTTLS command is expected to become the preferred technique. SMTP, IMAP, and POP3 all have established ports for use with TLS (port numbers 465, 993, and 995, respectively). When a client connects to one of these special ports (as configured in the Dispatcher configuration file), PMDF-TLS will immediately begin TLS negotiation. Once the negotiation is complete, the connection will be given to the service as usual.

[1] RFC 2487 defines the STARTTLS command for SMTP; RFC 2595 defines the STARTTLS command for IMAP and POP.

In the case that a STARTTLS command is used, the TCP connection is established on the usual port number (or an alternate port number if configured in the Dispatcher) and given to the service normally. For instance, if TLS is available to the client in an SMTP session, the server will advertise STARTTLS as one of its available SMTP extensions; the client will then issue the STARTTLS command, the server will acknowledge receipt of the SMTP command and instruct the client to begin TLS negotiation. Again, once the negotiation is complete, the connection continues normally.

15.2 Configuration

Configuration of TLS consists of two parts: setting up the certificate that will be used by PMDF-TLS, and enabling TLS functionality in the various PMDF servers.

15.2.1 Certificate Setup

Note: See the Glossary for definitions of unfamiliar terms.

PMDF-TLS requires a TLS *certificate* in order to accept TLS connections. This certificate is presented to the client during the negotiation of a TLS connection and is used to determine the secret *private key* that will be used to encrypt the connection between the server and the client.

Certificates can be requested from a *Certificate Authority* such as such as Thawte Consulting, Verisign, Inc., or a free certificate from Let's Encrypt.

It is possible to use self-signed certificates, but most clients and servers will no longer allow self-signed certificates, so they are not going to be covered here. If you want to create a self-signed certificate, the `openssl` utility on an OpenVMS or Linux system can be used to generate them.

15.2.1.1 Getting a Certificate Authority to Sign Your Certificate

Once you have your certificate request completed, you then need to have it signed by the Certificate Authority of your choice. Some sites can choose to have their requests signed by an in-house Authority, but many will choose to go to an independent Certificate Authority, such as Thawte Consulting (http://www.thawte.com/) or Verisign, Inc., (http://www.verisign.com/).

Both of these Authorities will provide complete information on what is needed to complete your certificate request. For PMDF usage, tell the signing Authority that you want a "web server" or "server" sort of certificate.

When you have finished the process of getting a signed certificate from a Certificate Authority, you'll have a new file that starts with

```
-----BEGIN CERTIFICATE-----
```

You should place this signed certificate file on your system as `server-pub.pem` in the PMDF table directory.

15.2.1.2 Chained Certificates

PMDF supports chained TLS certificates. In order to use these, concatenate all of the certificates into the `server-pub.pem` file in the PMDF table directory. The local server certificate should be first, followed by one or more intermediary certificates, and finally the root certificate. Make sure all of the separators (i.e. "——BEGIN CERTIFICATE—-") remain intact.

15.2.2 Enabling TLS Functionality in PMDF

In order to start using PMDF-TLS in conjunction with PMDF's servers, you must have your private key and public certificate ready and in place; these are normally obtained as described above in Section 15.2.1. Specifically, you must have the following files in the PMDF table directory:

- `server-priv.pem` (this file *must* be protected against world access; in addition, on UNIX it must be owned by the `pmdf` user), and
- `server-pub.pem`.

The `server-pub.pem` file should preferentially be a signed certificate returned by a well-recognized Certificate Authority in response to your certificate request, as described in Section 15.2.1.1 above.

PMDF's TLS-related services are either enabled by use of alternate port numbers or by turning on the STARTTLS functionality. The use of alternate port numbers is discussed further in Section 15.2.2.1 below. Configuring whether and when to use or allow the STARTTLS command in the PMDF SMTP server is discussed further in Section 15.2.2.2 below. PMDF's SASL use can also be combined with TLS use, as discussed in Section 15.2.2.3 below.

Note: As soon as the PMDF-TLS license is enabled, PMDF assumes that you want to use TLS. TLS is automatically initialized. This means, for example, that the STARTTLS functionality is automatically turned on for IMAP and POP server use on the standard port numbers.

15.2.2.1 Dispatcher-related Configuration for Alternate Port Numbers

If you have not already configured the Dispatcher, you must do so; see the appropriate edition of the *PMDF Installation Guide*. Once the Dispatcher is configured, you will have services defined in sections that look something like:

```
[SERVICE=SMTP]
PORT=25
...
```

To enable TLS for such a Dispatcher service, you simply add a TLS_PORT option to the configuration for that service. For example, to add TLS support on port 465 for SMTP (the established port for SMTP TLS use), you'd use:

```
[SERVICE=SMTP]
PORT=25
TLS_PORT=465
...
```

You can similarly add TLS support for POP3, IMAP, and HTTP by adding similar lines, *e.g.*,

```
[SERVICE=POP3]
PORT=110
TLS_PORT=995
...
!
[SERVICE=IMAP]
PORT=143
TLS_PORT=993
...
!
[SERVICE=HTTP]
PORT=7633
TLS_PORT=443
...
```

Note that to use TLS for HTTP, the user must specify `https://` in their browser when pointing to the PMDF web interface.

If you want to use multiple certificates, or if you want to specify use of a certificate stored in files with names other than the default `server-pub.pem` and `server-priv.pem` names, then you can set one or more instances of the TLS_CERTIFICATE Dispatcher option to specify your certificate(s). Up to five TLS_CERTIFICATE lines can be specified; each should specify a pair of files comprising a particular certificate.

Once the Dispatcher configuration modifications are complete, you must restart the Dispatcher (if it is currently running) or start it (if it is not currently running) so that the new Dispatcher configuration with the new port numbers takes effect; see Section 11.4.

15.2.2.2 TCP/IP Channel Configuration for TLS Use

PMDF supports a number of keywords on the TCP/IP channels to control whether TLS functionality is desired or required. The functions of these keywords are summarized in Table 15–1; see also Section 2.3.4.45 and Section 2.3.4.46.

Table 15–1 PMDF-TLS Channel Block keywords

Keyword	Usage
`notls`	The combination of `notlsserver` and `notlsclient`; this is the default
`maytls`	The combination of `maytlsserver` and `maytlsclient`
`musttls`	The combination of `musttlsserver` and `musttlsclient`
`notlsserver`	Do not offer the STARTTLS extension and do not accept a STARTTLS command from a remote client
`maytlsserver`	Offer and accept STARTTLS (if not already TLS-enabled)
`musttlsserver`	Offer and require STARTTLS (if not already TLS-enabled); if TLS has not been negotiated, refuse to accept any mail during this session with a "530" error
`notlsclient`	Do not attempt to use STARTTLS even if offered by a remote SMTP server
`maytlsclient`	If STARTTLS is offered by a remote SMTP server, attempt to use TLS
`musttlsclient`	Use STARTTLS if offered by a remote SMTP server, but if not available, this message delivery will be aborted
`tlsswitchchannel` *`channelname`*	If TLS is used, switch to the channel specified as the *`channelname`* parameter to this keyword
`nomsexchange`	Advertise (and accept) standard TLS commands; this is the default
`msexchange`	Advertise (and accept) broken TLS commands as used by, for instance, MS Exchange

Enabling (or requiring) the use of TLS can be of interest on dedicated channels intended for communicating sensitive information with companion systems that also support TLS.

Enabling the use of TLS for the SMTP server can also be of particular interest when SMTP SASL use has been enabled. Since with SMTP SASL use, a remote client will be sending a password over the network, then, especially if the PLAIN authentication mechanism is used (password sent "in the clear"), it can be particularly desirable to use TLS so that the entire transaction, including the password, is encrypted.

Use of the `tlsswitchchannel` keyword can be of interest for logging purposes, so that PMDF log entries show the message as coming in via a special channel. Use of the `tlsswitchchannel` keyword can also be of interest if it is desired to route messages submitted using TLS differently (using source channel specific rewrite rules) than messages submitted without TLS.

15.2.2.3 TLS Use and SASL

There is some added support for TLS in SASL. Using SASL, you can restrict which password methods are available based on the strength (or existence) of the encryption being used. This is controlled via the RESTRICT option in the PMDF security configuration file; see Section 14.2.2 for details. See also Example 15–5 below for an example of use of this option.

15.2.2.4 Sample TLS Configuration

A sample configuration of enabling TLS use in PMDF will be presented for a sample PMDF-TLS site domain.com. The significant features of this sample configuration are as follows.

1. The Dispatcher will be configured to automatically use TLS when connections come in on alternate, established ports for TLS use for SMTP, POP, and IMAP connections.
2. Incoming SMTP, POP, and IMAP connections will be categorized according as to whether the connection is coming from an "internal" source or an "external" source, where "internal" sources are assumed to be those in the 1.2.3.0 subnet.
3. POP and IMAP client connections from "external" sources will not be allowed to authenticate using the PLAIN or LOGIN mechanisms (which both involve sending the password "in the clear" over the network) unless at least 40 bits of TLS encryption are used by the client. That is, POP and IMAP clients connecting from "external" sources must either authenticate using the SASL CRAM-MD5 mechanism (or APOP in the case of POP), which involves encryption in the actual authentication mechanism, or must use (at least 40 bits of) TLS encryption for the overall transaction in order to use the PLAIN or LOGIN mechanisms for authentication.
4. TLS and SASL will be offered to incoming SMTP connections for clients that want to negotiate their use. "External" clients can not submit SMTP messages for relaying unless they use SASL to authenticate themselves. Further, when performing SASL authentication from "external" clients, PLAIN or LOGIN authentication (either of which involve sending the password "in the clear" over the network) will not be allowed unless at least 40 bits worth of TLS encryption are in use. That is, permitted SMTP message traffic will include:
   - Any messages submitted from "internal" clients.
   - Any messages submitted from "external" clients destined for "internal" recipients.
   - Any messages submitted from "external" clients that use SASL to authenticate themselves and TLS to encrypt that authentication (as well as the data transfer); this includes cases where the SASL authentication itself is performed using what is normally an "in the clear" mechanism such as PLAIN or LOGIN since TLS is being used to encrypt the overall message transaction.
   - Any messages submitted from "external" clients that use SASL to authenticate themselves using the CRAM-MD5 SASL mechanism.

In order to automatically allow TLS use when connecting to a TLS established alternate port, item (1) above, the Dispatcher configuration file includes TLS_PORT option settings as shown in Example 15–1 (OpenVMS) or Example 15–2 (UNIX) or Example 15–3 (Windows). Note that the samples shown are merely excerpts; a typical site is likely to have (and want) several additional services defined and several additonal options set within the individual service definitions.

Example 15–1 Excerpt of a Sample `dispatcher.cnf` File for TLS Ports on OpenVMS

```
!
! Global defaults
!
MIN_PROCS=1
MAX_PROCS=5
MIN_CONNS=3
MAX_CONNS=10
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=300
MAX_IDLE_TIME=600
!
! multithreaded SMTP server
!
[SERVICE=SMTP]
PORT=25
TLS_PORT=465
IMAGE=PMDF_EXE:TCP_SMTP_SERVER.EXE
LOGFILE=PMDF_LOG:TCP_SMTP_SERVER.LOG
!
! POP3 server
!
[SERVICE=POP3]
PORT=110
TLS_PORT=995
IMAGE=PMDF_EXE:POP3D.EXE
LOGFILE=PMDF_LOG:POP3D.LOG
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
TLS_PORT=993
IMAGE=PMDF_EXE:IMAPD.EXE
LOGFILE=PMDF_LOG:IMAPD.LOG
```

Example 15–2 Excerpt of a Sample `dispatcher.cnf` File for TLS Ports on UNIX

```
!
! Global defaults
!
MIN_PROCS=1
MAX_PROCS=5
MIN_CONNS=3
MAX_CONNS=10
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=300
MAX_IDLE_TIME=600
!
! multithreaded SMTP server
!
[SERVICE=SMTP]
PORT=25
TLS_PORT=465
IMAGE=/pmdf/bin/tcp_smtp_server
LOGFILE=/pmdf/log/tcp_smtp_server.log
!
! POP3 server
!
[SERVICE=POP3]
PORT=110
TLS_PORT=995
IMAGE=/pmdf/bin/pop3d
LOGFILE=/pmdf/log/pop3d.log
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
TLS_PORT=993
IMAGE=/pmdf/bin/imapd
LOGFILE=/pmdf/log/imapd.log
```

In order to categorize incoming SMTP, POP, and IMAP connections into those from "internal" sources (assumed to be any in the 1.2.3.0 subnet) and those from "external" sources, a PORT_ACCESS mapping table is used as shown in Example 15–4; see Section 11.5 for general background on the PORT_ACCESS mapping table and Section 14.3 for more details on categorizing incoming connections in particular.

In order to enforce that connections from "external" sources that perform authentication must either use an encrypted mechanism (*e.g.*, CRAM-MD5) for that authentication, or if using one of the PLAIN or LOGIN mechanisms must be securing the entire transaction using TLS with at least 40 bits of encryption, item (3) and part of item (4) above, a PMDF security configuration file as in Example 15–5 is used. Note that this security configuration file uses different security rulesets for "internal" and "external" connections, as selected by the PORT_ACCESS mapping shown in Example 15–4 above.

Example 15–3 Excerpt of a Sample `dispatcher.cnf` File for TLS Ports on NT

```
!
! Global defaults
!
MIN_PROCS=1
MAX_PROCS=5
MIN_CONNS=3
MAX_CONNS=10
MAX_SHUTDOWN=2
MAX_LIFE_TIME=86400
MAX_LIFE_CONNS=300
MAX_IDLE_TIME=600
!
! multithreaded SMTP server
!
[SERVICE=SMTP]
PORT=25
TLS_PORT=465
IMAGE=C:\pmdf\bin\tcp_smtp_server
LOGFILE=C:\pmdf\log\tcp_smtp_server.log
!
! POP3 server
!
[SERVICE=POP3]
PORT=110
TLS_PORT=995
IMAGE=C:\pmdf\bin\pop3d
LOGFILE=C:\pmdf\log\pop3d.log
!
! IMAP server
!
[SERVICE=IMAP]
PORT=143
TLS_PORT=993
IMAGE=C:\pmdf\bin\imapd
LOGFILE=C:\pmdf\log\imapd.log
```

Example 15–4 Sample PORT_ACCESS Mapping Categorizing Incoming Connections

```
PORT_ACCESS
! Connections from the 1.2.3.0 subnet will be considered internal
!
  TCP|*|*|$(1.2.3.0/24)|*   $YINTERNAL
!
! Block all access to the PMDF HTTP server from external sources
!
  TCP|7633|*|*              $N
!
! All other connections will be considered external
!
  TCP|*|*|*|*               $YEXTERNAL
```

In order to restrict SMTP relaying from “external” sources unless they authenticate themselves using SASL, a TCP/IP channel configuration similar to that shown in

Example 15–5 Sample `security.cnf` File for Enforcing TLS Use with PLAIN and LOGIN Mechanisms for External Connections

```
[RULESET=DEFAULT]
ENABLE=MSGSTORE/*,PASSDB/*,SYSTEM/*
!
[RULESET=INTERNAL]
ENABLE=MSGSTORE/*,PASSDB/*,SYSTEM/*,LOGIN/LOGIN
!
[RULESET=EXTERNAL]
ENABLE=MSGSTORE/*,PASSDB/*,SYSTEM/*,LOGIN/LOGIN
RESTRICT=PLAIN:40,LOGIN:40
```

Section 14.4 will be used. However, the configuration will be further tuned to allow TLS use for incoming connections. The TLS use is expected to be particularly relevant for connections from "external" sources since, due to the PORT_ACCESS mapping and PMDF security configuration file described above, "external" clients that want to SASL authenticate themselves using either of the PLAIN or LOGIN mechanisms can only do so if they also use TLS. Appropriate rewrite rules in the PMDF configuration file could be along the lines of:

```
.domain.com     $U%$H$D@TCP-INTERNAL
[1.2.3.]      $U%[1.2.3.$L]@TCP-INTERNAL$E$R
```

with channel definitions along the lines of

```
tcp_local smtp mx single_sys maytls \
  maysaslserver saslswitchchannel tcp_auth switchchannel
TCP-DAEMON

tcp_internal smtp mx single_sys maytls \
  maysaslserver allowswitchchannel routelocal
TCP-INTERNAL

tcp_auth smtp mx single_sys maytlsserver \
  mustsaslserver noswitchchannel
TCP-AUTH
```

and with an ORIG_SEND_ACCESS mapping table along the lines of:

```
ORIG_SEND_ACCESS

  tcp_local|*|tcp_local|*   $NSMTP$ relaying$ not$ permitted
```

15.3 Recording of TLS Use in Received: Headers and PMDF Log Entries

When a message is received with TLS, the Received: header PMDF constructs will include the words "with ESMTPS" (if received with TLS only) or "with ESMTPSA" (if received with TLS and SASL) rather than the usual "with SMTP" (received without extended SMTP), "with ESMTP" (received with extended SMTP), or "with ESMTPA" (received with SASL but not TLS).

If the `logging` channel keyword is enabled, then messages received or sent with TLS used will show an “S” (security) character in addition to the usual “E” or “D” character.

16 Mail Filtering and Access Control

A common goal is to outright reject messages from (or to) certain users at the system level, or to institute more complex restrictions of message traffic between certain users, or to allow users to set up filters on their own incoming messages (including rejecting messages based on contents of the message headers). This chapter will discuss some of PMDF's facilities in these areas, including: system level mapping tables such as SEND_ACCESS, FROM_ACCESS, and MAIL_ACCESS that permit both simple and sophisticated restrictions of message traffic based on source and destination and envelope From: and To: addresses—see Section 16.1; user level (and system level) message filtering using Sieve, including sophisticated filtering based on message headers.

Related topics discussed elsewhere in this manual include: system level blocking of connections from (or to) particular systems—see the discussion of the PORT_ACCESS mapping table in Section 11.5; using different authentication mechanisms for different sorts of connections – see Chapter 14; and techniques falling under the general category of protecting against denial of service attacks—see Section 28.4.5.3.

Use of mapping tables such as SEND_ACCESS, MAIL_ACCESS, FROM_ACCESS, *etc.*, is an efficient approach when "envelope level" controls are desired—see Section 16.1. When users want to implement their own personalized controls, or when header-based filtering is desired, the more general mail filtering approach using Sieve is likely appropriate—see Section 16.2.

16.1 Address-based Access Control Mappings

There are several mapping tables that can be used to control who can or can not send mail, receive mail, or both. For general information on the format and usage of the PMDF mapping file, see Chapter 5. The SEND_ACCESS, ORIG_SEND_ACCESS, MAIL_ACCESS, ORIG_MAIL_ACCESS, and FROM_ACCESS mappings are described below. The nature of these mappings is very general and allows per channel granularity.

In the case of messages that come in channels where the underlying network connection is handled via the PMDF Dispatcher, including the PMDF multithreaded TCP SMTP channels, and the Lotus Notes channels, there is a related mapping table, PORT_ACCESS, which can be used to block incoming connections based on IP number; see Section 21.2.1. Although the PORT_ACCESS mapping table does not allow for the fine level of granularity of the SEND_ACCESS and related mapping tables and applies only to certain channels, it is more efficient for what it does do, since it rejects a disallowed hosts' connection attempt at the TCP level, before the channel dialogue has even begun.

The MAIL_ACCESS and ORIG_MAIL_ACCESS mappings are the most general, having available not only the address and channel information available to SEND_ACCESS and ORIG_SEND_ACCESS, but also any information that would be available via the PORT_ACCESS mapping table, including IP address and port number information.

16.1.1 The SEND_ACCESS and ORIG_SEND_ACCESS Mappings

The SEND_ACCESS and ORIG_SEND_ACCESS mapping tables can be used to control who can or can not send mail, receive mail, or both. The access checks have available by default:

- a message's envelope From: address
- a message's envelope To: address
- what channel the message came in on
- what channel the message would attempt to go out on

If the ACCESS_ORCPT option is specified as 1 in the PMDF option file, then a fifth piece of information is added to the probe: the original recipient information (the value of the ORCPT option in the SMTP protocol exchange).

Note that when the To: addresses are irrelevant and only the From: address matters, then use of the FROM_ACCESS mapping table, described below in Section 16.1.3, can be more convenient and efficient.

If a SEND_ACCESS or ORIG_SEND_ACCESS mapping table exists, then for each recipient of every message passing through PMDF, PMDF will probe the table by default with a probe string of the form (note the use of the vertical bar character, |):

src-channel|*from-address*|*dst-channel*|*to-address*

If the ACCESS_ORCPT option is set to 1, then the probe string form is:

src-channel|*from-address*|*dst-channel*|*to-address*|*orcpt-address*

where

- *src-channel* is the channel originating the message (*i.e.,* queueing the message);
- *from-address* is the address of the message's originator;
- *dst-channel* is the channel to which the message will be queued;
- *to-address* is the address to which the message is addressed;
- *orcpt-address* is the original recipient address (ORCPT).

The use of an asterisk in any of these fields causes that field to match any channel or address, as appropriate.

The addresses here are envelope addresses, that is, envelope From: address and envelope To: address. In the case of SEND_ACCESS, the envelope To: address is checked after rewriting, alias expansion, *etc.*, have been performed; in the case of ORIG_SEND_ACCESS the originally specified envelope To: address is checked after rewriting, but before alias expansion.

If the probe string matches a pattern (*i.e.,* the left hand side of an entry in the table), then the resulting output of the mapping is checked. If the output contains the flags $Y or $y, then the enqueue for that particular To: address is permitted. If the mapping output contains any of the flags $N, $n, $F, or $f, then the enqueue to that particular address is rejected. In the case of a rejection, optional rejection text can be supplied in the mapping output. This string will be included in the rejection error PMDF issues.[1] If no string is output (other than the $N, $n, $F, or $f flag), then default rejection text will be used. See Table 16–1 for descriptions of additional flags.

In the following example, note that mail sent from OpenVMS user agents such as VMS MAIL, PMDF MAIL, *etc.*, or from UNIX user agents such as mail, Pine, *etc.*, originates from the local, l, channel and messages to the Internet go out a TCP/IP channel of some sort. Now, suppose that local users, with the exception of the postmaster, are not allowed to send mail to the Internet but can receive mail from there. Then the SEND_ACCESS mapping table shown in Example 16–1 is one possible way to enforce this restriction. In that example, the local host name is assumed to be example.com. In the channel name "tcp_*", a wild card is used so as to match any possible TCP/IP channel name (*e.g.,* tcp_local, tcp_gateway, *etc.*). In the rejection message, dollar signs are used to quote spaces in the message. Without those dollar signs, the rejection would be ended prematurely and only read "Internet" instead of "Internet postings are not permitted". Note that this example ignores other possible sources of "local" postings such as from PC based mail systems or POP or IMAP clients.

Example 16–1 Restricting Internet Mail Access

```
SEND_ACCESS

  *|postmaster@example.com|*|*  $Y
  *|*|*|postmaster@example.com  $Y
  l|*@example.com|tcp_*|*       $NInternet$ postings$ are$ not$ permitted
```

Table 16–1 Access Mapping Flags®

Flag	Description
$B	Redirect the message to the bitbucket
$H	Hold the message as a .HELD file
$Y	Allow access
	Flags with arguments, in argument reading order†
$J*address*	Replace original envelope From: address with specified *address*§
$K*address*	Replace original Sender: address with specified *address*§
$I*user*|*identifier*	Check specified user for specified identifier (OpenVMS) or groupid (UNIX)

§Available for FROM_ACCESS table only

†To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

[1] Note that it is up to whatever is attempting to send the message whether the PMDF rejection error text is actually presented to the user who attempted to send the message. In particular, in the case when SEND_ACCESS is used to reject an incoming SMTP message, PMDF merely issues an SMTP rejection code including the optional rejection text; it is up to the sending SMTP client to use that information to construct a bounce message to send back to the original sender.

Table 16–1 (Cont.) Access Mapping Flags®

Flag	Description
	Flags with arguments, in argument reading order†
$<*string*	Send *string* as an OPCOM broadcast (OpenVMS) or to syslog (UNIX) or to the event log (NT) if probe matches‡
$>*string*	Send *string* as an OPCOM broadcast (OpenVMS) or to syslog (UNIX) or to the event log (NT) if access is rejected ‡
$D*delay*	Delay response for an interval of *delay* hundredths of seconds; a positive value causes the delay to be imposed on each command in the transaction; a negative value causes the delay to be imposed only on the address handover (SMTP MAIL FROM: command for the FROM_ACCESS table; SMTP RCPT TO: command for the other tables)
$T*tag*	Prefix with tag *tag*
$A*header*	Add the header line *header* to the message
$X*error-code*	Issue the specified *error-code* extended SMTP error code if rejecting the message
$N*string*	Reject access with the optional error text *string*
$F*string*	Synonym for $N*string*, *i.e.*, reject access with the optional error text *string*

†To use multiple flags with arguments, separate the arguments with the vertical bar character, |, placing the arguments in the order listed in this table.

‡It is a good idea to use the $D flag when dealing with problem senders, to prevent a denial of service attack. In particular, it is a good idea to use $D in any $> entry or $< entry rejecting access.

16.1.2 The MAIL_ACCESS and ORIG_MAIL_ACCESS Mappings

The MAIL_ACCESS mapping table is a superset of the SEND_ACCESS and PORT_ACCESS mapping tables; that is, it combines both the channel and address information of SEND_ACCESS, with the IP address and port number information of PORT_ACCESS. Similarly, the ORIG_MAIL_ACCESS mapping table is a superset of the ORIG_SEND_ACCESS and PORT_ACCESS mapping tables. The format for the probe string for MAIL_ACCESS is

port_access-probe-info|*app-info*|*submit-type*|*send_access-probe-info*

and similarly the format for the probe string for ORIG_MAIL_ACCESS is

port_access-probe-info|*app-info*|*submit-type*|*orig_send_access-probe-info*

Here *port_access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe in the case of incoming SMTP messages, or will be blank otherwise, and *app-info* will usually be SMTP in the case of messages submitted via SMTP, and blank otherwise. *submit-type* can be one of MAIL, SEND, SAML, or SOML, corresponding to how the message was submitted into PMDF. Normally the value is MAIL, meaning it was submitted as a message; SEND, SAML, or SOML can occur in the case of broadcast requests (or combined broadcast/message requests) submitted

to the SMTP server. And for the MAIL_ACCESS mapping, *send_access-probe-info* consists of all the information usually included in a SEND_ACCESS mapping table probe. Similarly for the ORIG_MAIL_ACCESS mapping, *orig_send_access-probe-info* consists of all the information usually included in an ORIG_SEND_ACCESS mapping table probe.

Having the incoming TCP/IP connection information available in the same mapping table as the channel and address information makes it more convenient to impose certain sorts of controls, such as enforcing what envelope From: addresses are allowed to appear in messages from particular IP addresses. This can be desirable to limit cases of e-mail forgery, or to encourage users to configure their POP and IMAP clients' From: address appropriately. For instance, a site that wants to allow the envelope From: address vip@ourcorp.com to appear only on messages coming from the IP address 1.2.3.1 and 1.2.3.2, and to ensure that the envelope From: addresses on messages from any systems in the 1.2.0.0 subnet are from ourcorp.com, might use a MAIL_ACCESS mapping table along the lines shown in Example 16–2.

Example 16–2 Enforcing Use of Proper Source Addresses

```
MAIL_ACCESS

! Entries for vip's two systems
!
  TCP|*|25|1.2.3.1|*|SMTP|MAIL|tcp_*|vip@ourcorp.com|*|*  $Y
  TCP|*|25|1.2.3.2|*|SMTP|MAIL|tcp_*|vip@ourcorp.com|*|*  $Y
!
! Disallow attempts to use vip's From: address from other systems
!
  TCP|*|25|*|*|SMTP|MAIL|tcp_*|vip@ourcorp.com|*|*  \
          $N500$ Not$ authorized$ to$ use$ this$ From:$ address
!
! Allow sending from within our subnet with ourcorp.com From: addresses
!
  TCP|*|25|1.2.*.*|*|SMTP|MAIL|tcp_*|*@ourcorp.com|*|*  $Y
!
! Allow notifications through
!
  TCP|*|25|1.2.*.*|*|SMTP|MAIL|tcp_*||*|*  $Y
!
! Block sending from within our subnet with non-ourcorp.com addresses
!
  TCP|*|25|1.2.*.*|*|SMTP|MAIL|tcp_*|*|*|*  \
          $NOnly$ ourcorp.com$ From:$ addresses$ authorized
```

16.1.3 The FROM_ACCESS Mapping Table

The FROM_ACCESS mapping table can be used to control who can send mail, or to override purported From: addresses with authenticated addresses, or both.

The input probe string to the FROM_ACCESS mapping table is similar to that for a MAIL_ACCESS mapping table, minus the destination channel and address, and with the addition of authenticated sender information, if available. Thus if a FROM_ACCESS mapping table exists, then for each attempted message submission PMDF will probe the table with a probe string of the form (note the use of the vertical bar character, |)

port_access-probe-info|*app-info*|*submit-type*|*src-channel*|*from-address*|*auth-from*

Here *port_access-probe-info* consists of all the information usually included in a PORT_ACCESS mapping table probe in the case of incoming SMTP messages, or will be blank otherwise, and *app-info* will usually be SMTP in the case of messages submitted via SMTP, and blank otherwise. Note that if you are using TLS, that *app-info* contains the TLS information, for example `SMTP/TLS-168-DES-CBC3-SHA`, so we recommend you use SMTP*.

submit-type can be one of MAIL, SEND, SAML, or SOML, corresponding to how the message was submitted into PMDF. Normally the value is MAIL, meaning it was submitted as a message; SEND, SAML, or SOML can occur in the case of broadcast requests (or combined broadcast/message requests) submitted to the SMTP server. *src-channel* is the channel originating the message (*i.e.,* queueing the message); *from-address* is the address of the message's purported originator; and *auth-from* is the authenticated originator address, if such information is available, or blank if no authenticated information is available.

If the probe string matches a pattern (*i.e.,* the left hand side of an entry in the table), then the resulting output of the mapping is checked. If the output contains the flags $Y or $y, then the enqueue for that particular To: address is permitted. If the mapping output contains any of the flags $N, $n, $F, or $f, then the enqueue to that particular address is rejected. In the case of a rejection, optional rejection text can be supplied in the mapping output. This string will be included in the rejection error PMDF issues.[2] If no string is output (other than the $N, $n, $F, or $f flag), then default rejection text will be used. See Table 16–1 for descriptions of additional flags.

Besides determining whether to allow a message to be submitted based on the originator, FROM_ACCESS can also be used to alter the envelope From: address via the `$J` flag, or to modify the effect of the `authrewrite` channel keyword (adding a Sender: header address on an accepted message) via the `$K` flag. For instance, this mapping table can be used to cause the original envelope From: address to simply be replaced by the authenticated address:

[2] Note that it is up to whatever is attempting to send the message whether the PMDF rejection error text is actually presented to the user who attempted to send the message. In particular, in the case when FROM_ACCESS is used to reject an incoming SMTP message, PMDF merely issues an SMTP rejection code including the optional rejection text; it is up to the sending SMTP client to use that information to construct a bounce message to send back to the original sender.

```
FROM_ACCESS

  *|SMTP*|*|tcp_local|*|      $Y
  *|SMTP*|*|tcp_local|*|*     $Y$J$4
```

When using the FROM_ACCESS mapping table to modify the effect on having `authrewrite` set to a nonzero value on some source channel, it is not necessary to use FROM_ACCESS if the authenticated address is going to be used verbatim. For instance, with `authrewrite 2` set on the tcp_local channel, the following FROM_ACCESS mapping table would not be necessary:

```
FROM_ACCESS

  *|SMTP*|*|tcp_local|*|      $Y
  *|SMTP*|*|tcp_local|*|*     $Y$K$4
```

as `authrewrite` alone is sufficient to get this effect (adding the authenticated address verbatim). However, the real purpose of FROM_ACCESS is to permit more complex and subtle alterations. For instance, perhaps you want to force on a Sender: header only in cases where the authenticated address differs from the envelope From: address, with subaddresses *not* being considered to constitute a difference, as illustrated in the following table:

```
FROM_ACCESS

! If no authenticated address is available, do nothing
  *|SMTP*|*|tcp_local|*|              $Y
! If authenticated address matches envelope From:, do nothing
  *|SMTP*|*|tcp_local|*|$3*          $Y
! If authenticated address matches envelope From: sans subaddress, do nothing
  *|SMTP*|*|tcp_local|*+*@*|$3*@$5*   $Y
! Fall though to...
! ...authenticated address present, but didn't match, so force Sender: header
  *|SMTP*|*|tcp_local|*|*             $Y$ASender:$ $4
```

16.1.4 When Access Controls are Applied

PMDF checks access control mappings as early as possible. Exactly when this happens depends upon the e-mail protocol in use—when the information that must be checked becomes available. But for instance in the case of the SMTP protocol, where addresses are presented in the initial part of the attempted message handover, well before the message data itself would be handed over, note that a FROM_ACCESS rejection will occur in response to the MAIL FROM: command, *before* the sending side ever gets to send the recipient information let alone the message data, while a SEND_ACCESS or MAIL_ACCESS sort of rejection will occur in response to the RCPT TO: command, *before* the sending side ever gets to send the message data. If an SMTP message is rejected, PMDF never even accepts or sees the message data, thus minimizing the overhead of performing such rejections.

If multiple access control mapping tables exist, PMDF will check them all; that is, a FROM_ACCESS, a SEND_ACCESS mapping table, an ORIG_SEND_ACCESS mappings table, a MAIL_ACCESS mapping table, and an ORIG_MAIL_ACCESS mapping table can all be in effect.

16.1.5 Testing Access Control Mappings

The PMDF TEST/REWRITE (OpenVMS) or `pmdf test -rewrite` (UNIX and NT) utility, particularly with the /FROM, /SOURCE_CHANNEL, and /DESTINATION_CHANNEL (OpenVMS) or `-from`, `-source_channel`, and `-destination_channel` (UNIX and NT) qualifiers, can be useful in testing access control mappings. For instance, a SEND_ACCESS mapping table of

```
SEND_ACCESS

  tcp_local|friendly@somewhere.com|l|AdamUser@example.com      $Y
  tcp_local|unwelcome@elsewhere.com|l|AdamUser@example.com     $NGo$ away!
```

can be probed as follows:

```
$ PMDF TEST/REWRITE/FROM="friendly@somewhere.com" -
_$  /SOURCE=tcp_local/DESTINATION=l AdamUser@example.com
...
Submitted address list:
  l
    AdamUser (EXAMPLE.COM) *NOTIFY FAILURES* *NOTIFY DELAYS*

Submitted notifications list:

$ PMDF TEST/REWRITE/FROM="unwelcome@elsewhere.com" -
_$  /SOURCE=tcp_local/DESTINATION=l AdamUser@example.com
...
Submitted address list:
Address list error -- Go away!: AdamUser@example.com

Submitted notifications list:
```

16.1.6 SMTP Relay Blocking

One application of access control mappings is to prevent people from relaying SMTP mail through your PMDF system; for instance, to prevent people from using your PMDF system to relay junk mail to hundreds or thousands of Internet mail boxes.

By default PMDF does not prevent SMTP relaying activity: for starters, SMTP relaying is not necessarily a bad thing. Sites should only block such activity if it is causing them difficulty. Morever, note that local users using POP or IMAP depend upon PMDF to act as an SMTP relay. Blocking unauthorized relaying while allowing it for legitimate local users requires configuring PMDF to know how to distinguish between the two classes of users. Configuring PMDF to make this distinction is the topic of the next section.

16.1.6.1 Differentiating Between Internal and External Mail

In order to block mail relaying activities, you must first be able to differentiate between "internal" mail originated at your site and "external" mail originated out on the Internet and passing through your system back out to the Internet. The former class of mail you want to permit; the latter class you want to block. This differentiation is achieved using the `switchchannel` keyword on your inbound SMTP channel, usually the tcp_local channel.

The `switchchannel` keyword works by causing the PMDF SMTP server to look at the actual IP address associated with the incoming SMTP connection. PMDF uses that IP address, in conjunction with your rewrite rules, to differentiate between an SMTP connection originated within your domain and a connection from outside of your domain. This information can then be used to segregate the message traffic between internal and external traffic.[3]

Let's now actually change your PMDF configuration so that you can differentiate between your internal and external message traffic. This is done by editing your PMDF configuration file, `pmdf.cnf`, located in the PMDF table directory.

1. In your configuration file, locate your local channel; this is the l (lowercase L) channel on OpenVMS or UNIX, or the msgstore channel on NT. It is usually preceded by the first blank line to appear in the configuration file. Immediately before the local channel, add the following defaults channel:

   ```
   defaults noswitchchannel
   ```

   So that the configuration file appears something like

   ```
   ! final rewrite rules

   defaults noswitchchannel

   ! Local channel
   l ...
   ```

   If you already have a defaults channel at this point in your configuration file, then simply add the `noswitchchannel` keyword to it.

2. Next, modify your incoming TCP/IP channel to specify the `switchchannel` and `remotehost` keywords, *e.g.*,

   ```
   tcp_local smtp single_sys mx switchchannel remotehost
   TCP-DAEMON
   ```

[3] Since the source IP address is used, it is important that your network router connecting your local network to the Internet be configured to prevent "IP spoofing". Contact your router vendor for assistance if you are unsure of how to configure your router.

3. Then, after your incoming TCP/IP channel definition, add a similar new channel but with a different name, *e.g.*:)

```
tcp_auth smtp single_sys mx allowswitchchannel routelocal
TCP-INTERNAL
```

The `routelocal` channel keyword is used to cause short-circuited rewriting of source routed addresses through this channel, thereby blocking possible attempts to relay by means of looping through internal SMTP hosts via explicitly source routed addresses.

4. Finally, add rewrite rules to route hosts and IP addresses in your domain to the tcp_internal channel. For instance, if your domain name is example.com and your domain's IP numbers are in the [a.b.subnet] range, then add to the top of your configuration file the rewrite rules

```
.example.com     $U%$H$D@TCP-INTERNAL
[a.b.]           $U%[a.b.$L]@TCP-INTERNAL$E$R
```

With the above configuration changes, SMTP mail generated within your domain will come in via the tcp_internal channel. All other SMTP mail will come in via the tcp_local channel. You can therefore distinguish between internal and external mail based upon which channel it comes in on.

How does the above work? The key is the `switchchannel` keyword. In Step 2 above, that keyword is applied to the tcp_local channel. When a message comes in your SMTP server, that keyword causes the server to look at the source IP address associated with the incoming connection. The server attempts a reverse-pointing envelope rewrite of the literal IP address of the incoming connection, looking for an associated channel. If that rewrite matches a local host of yours, then the rewrite rules added in Step 4 cause the address to rewrite to the tcp_internal channel added in Step 3. Since the tcp_internal channel is marked with the `allowswitchchannel` keyword, the message is "switched" to the tcp_internal channel and comes in on that channel. If the message comes in from an external source, the IP address will not correspond to an internal source. In that case the reverse-pointing envelope rewrite will either rewrite to the tcp_local channel or to some other channel. However, it will not rewrite to the tcp_internal channel and since all other channels were marked `noswitchchannel` in Step 1, the message will not "switch" to another channel and will remain with the tcp_local channel.

Note: Note that any mapping table or conversion file entries which use the string "tcp_local" can need to be changed to either "tcp_*" or "tcp_internal" depending upon the usage.

16.1.6.2 Differentiating Authenticated Users' Mail

Sometimes administratively "local" users of IMAP and POP clients can want to submit their messages when they aren't physically local (aren't physically on your network). Though such message submissions come in from an external IP address—for instance, arbitrary Internet Service Providers—if your users use mail clients that can perform SASL authentication, then their authenticated connections can be distinguished from arbitrary other external connections. The authenticated submissions you can then permit, while denying non-authenticated relay submission attempts.

Differentiating between authenticated and non-authenticated connections is achieved using the `saslswitchchannel` keyword on your inbound SMTP channel, usually the tcp_local channel.

The `saslswitchchannel` keyword takes an argument specifying the channel to switch to; if an SMTP sender succeeds in authenticating, then their submitted messages are considered to come in the specified switched to channel.

We will now continue the example of the previous section, adding in distinguishing authenticated submissions.

5. In your configuration file, add a new TCP/IP channel definition with a distinct name, *e.g.*:

```
tcp_auth smtp single_sys mx mustsaslserver noswitchchannel
TCP-AUTH
```

This channel should not allow regular channel switching (that is, it should have `noswitchchannel` on it either explicitly or implied by a prior `defaults` line). This channel should have `mustsaslserver` on it.

6. Modify your tcp_local channel by adding `maysaslserver` and `saslswitchchannel tcp_auth` to it. So your tcp_local channel definition should now appear along the lines of:

```
tcp_local smtp mx single_sys maysaslserver saslswitchchannel tcp_auth \
  switchchannel
TCP-DAEMON
```

With this configuration, SMTP mail sent by users who can authenticate with a local password will now come in the tcp_auth channel. Unauthenticated SMTP mail sent from internal hosts will still come in tcp_internal. All other SMTP mail will come in tcp_local.

16.1.6.3 Preventing Mail Relaying

Now to the point of this example: preventing unauthorized people from relaying SMTP mail through your system. First, keep in mind that you want to allow local users to relay SMTP mail. For instance, POP and IMAP users rely upon using PMDF to send their mail. Note that local users can either be physically local, in which case their messages come in from an internal IP address, or can be physically remote but able to authenticate themselves as “local” users. It’s random people out on the Internet who you want to prevent from using you as a relay. With the configuration from Section 16.1.6.1 and Section 16.1.6.2, you can differentiate between these classes of users and block the correct class. Specifically, you want to block mail from coming in your tcp_local channel and going back out that same channel. To that end, an ORIG_SEND_ACCESS mapping table is used.

An ORIG_SEND_ACCESS mapping table can be used to block traffic based upon the source and destination channel. In this case, traffic from and back to the tcp_local channel is to be blocked. This is realized with the following ORIG_SEND_ACCESS mapping table:

```
ORIG_SEND_ACCESS
  tcp_local|*|tcp_local|*                $NRelaying$ not$ permitted
```

In the above, the entry states that messages cannot come in the tcp_local channel and go right back out it. That is, this entry disallows external mail from coming in your SMTP server and being relayed right back out to the Internet.

An ORIG_SEND_ACCESS mapping table is used rather than a SEND_ACCESS mapping table, so that the blocking will not apply to addresses that originally match the l (lowercase "L") channel (but which can expand via an alias or mailing list definition back to an "external" address). With a SEND_ACCESS mapping table one would have to go to extra lengths to allow outsiders to send to mailing lists that expand back out to "external" users, or to send to users who forward their messages back out to "external" addresses.

16.1.6.4 Allowing localhost Submissions to the SMTP Port

This section discusses a further refinement of the SMTP relay blocking approach described above.

Some sites can want to disallow submissions to the SMTP port from clients running on the system itself; for instance, if no legitimate applications are expected to attempt this, then blocking such submissions closes one avenue by which users can spoof (forge) e-mail.

Other sites, however, can have legitimate applications that perform message submission by making a TCP/IP connection to the SMTP port of their own system. For instance, some third party mailing list expansion applications (though not PMDF's own mailing list expansion) operate in this way.

Further, to simplify their configuration such applications can connect using a loopback name or address, such as LOCALHOST or [127.0.0.1], rather than using the system's domain name. Depending on the underlying TCP/IP package, this can result in the incoming connection also appearing to come "from" LOCALHOST or [127.0.0.1], rather than coming from the system's specific domain name or IP address.

Using merely the internal *vs* external differentiation as shown above in Section 16.1.6.1 would result in treating SMTP submissions from clients on the host system itself as coming in the tcp_local channel. Thus if tcp_local to tcp_local message traffic is prevented, as discussed in Section 16.1.6.3, then any users or applications attempting to submit messages in such a way would not be allowed to submit messages to external addressees.

If you want to treat such submissions as "internal" submissions, for instance, in order to allow third party mailing list applications to operate even if SMTP relay blocking has been implemented, then an additional configuration step should be performed.

To the very top of the PMDF configuration file, add rewrite rules for the local host name (or LOCALHOST or [127.0.0.1], depending on from where the underlying TCP/IP package "sees" the connection as having originated) of the following form:

```
localhostname           $U%localhostname@TCP-INTERNAL$E$R
[localhostipnumber]     $U%localhostname@TCP-INTERNAL$E$R
LOCALHOST               $U%localhostname@TCP-INTERNAL$E$R
[127.0.0.1]             $U%localhostname@TCP-INTERNAL$E$R
```

where *localhostname* is the official host name on the L channel. The $E and $R control sequences on these rewrite rules, which limit their effect to envelope From: address rewriting, mean that normal rewriting will still apply to addresses addressed to the local system. Yet `switchchannel` rewriting will use these rules also, and hence will see message submissions from the system to its own SMTP port as "internal" submissions.

Warning: It is not recommended to add this rewrite rule for LOCALHOST unless you absolutely require it, as it opens up your PMDF implementation to abuse by spammers who can easily set up their DNS entries to say that their name is "localhost". If at all possible, do rewriting based solely on IP addresses rather than DNS names, as IP addresses are much harder to forge.

16.1.7 Efficiently Handling Large Numbers of Access Entries

Sites that use very large numbers of entries in mapping tables should consider organizing their mapping tables to have a few general wildcarded entries that call out to the general database for the specific lookups. It is much more efficient to have a few mapping table entries calling out to the general database for specific lookups than to have huge numbers of entries directly in the mapping table.

One case in particular is that some sites like to have per user controls on who can send and receive Internet e-mail. Such controls are conveniently implemented using an access mapping table such as ORIG_SEND_ACCESS. For such uses, efficiency and performance can be greatly improved by storing the bulk of the specific information (*e.g.*, specific addresses) in the general database with mapping table entries structured to call out appropriately to the general database.

For instance, consider a mapping table:

```
ORIG_SEND_ACCESS
```

```
! Users allowed to send to Internet
!
  *|adam@domain.com|*|tcp_local    $Y
  *|betty@domain.com|*|tcp_local   $Y
! ...etc...
!
! Users not allowed to send to Internet
!
  *|norman@domain.com|*|tcp_local  $NInternet$ access$ not$ permitted
  *|opal@domain.com|*|tcp_local    $NInternet$ access$ not$ permitted
! ...etc...
!
! Users allowed to receive from the Internet
!
  tcp_*|*|*|adam@domain.com        $Y
  tcp_*|*|*|betty@domain.com       $Y
! ...etc...
!
! Users not allowed to receive from the Internet
!
  tcp_*|*|*|norman@domain.com      $NInternet$ e-mail$ not$ accepted
  tcp_*|*|*|opal@domain.com        $NInternet$ e-mail$ not$ accepted
! ...etc...
```

Rather than using such a mapping table with each user individually entered into the table, a more efficient setup (much more efficient if hundreds or thousands of user entries are involved) would be as follows. Use general database entries of, say:

```
SEND|adam@domain.com    $Y
SEND|betty@domain.com   $Y
! ...etc...
SEND|norman@domain.com  $NInternet$ access$ not$ permitted
SEND|opal@domain.com    $NInternet$ access$ not$ permitted
! ...etc...
RECV|adam@domain.com    $Y
RECV|betty@domain.com   $Y
! ...etc...
RECV|norman@domain.com  $NInternet$ e-mail$ not$ accepted
RECV|opal@domain.com    $NInternet$ e-mail$ not$ accepted
```

with an ORIG_SEND_ACCESS mapping table of:

```
ORIG_SEND_ACCESS

! Check if may send to Internet
!
  *|*|*|tcp_local       $C${SEND|$1}$E
!
! Check if may receive from Internet
!
  tcp_*|*|*|*           $C${RECV|$3}$E
```

Here the use of the arbitrary strings “SEND | ” and “RECV | ” in the general database left hand sides (and hence in the general database probes generated by the mapping table) provides a way to distinguish between the two sorts of probes being made. The wrapping of the general database probes with the $C and $E flags, as shown, is typical of mapping table callouts to the general database; see Section 5.3.2.8 for an additional

discussion. For a discussion of the general database itself – where it is located and how to build it—see Section 2.2.6.5.

The above example showed a case of simple mapping table probes getting checked against general database entries. Mapping tables with much more complex probes can also benefit from use of the general database.

16.1.8 DNS_VERIFY

DNS_VERIFY can be used to validate domain names or IP addresses via DNS. For example, it can be used to verify that an entry in DNS exists for the domain used in the SMTP `MAIL FROM:` command, or to look up an IP address in the blackhole lists supplied by such services as MAPS and ORBS. The message can be rejected or accepted based on the presence or absence of a corresponding DNS record, or a new header can be added to the message to indicate the problem.

Note: Performing DNS checks may result in the rejection of some valid messages. For instance, this could include mail from legitimate sites that simply have not yet registered their domain name, or during periods of bad information in DNS.

Please be aware that doing DNS lookups is contrary to the spirit of being generous in what you accept, as expressed in RFC 1123, Requirements for Internet Hosts. However, some sites may desire to perform such checks in cases where junk e-mail (SPAM) is, for example, being sent with forged e-mail addresses from non-existent domains.

If DNS or connections to the sites being used for DNS verification become unavailable then mail delivery will be impacted. Use of DNS_VERIFY can impact performance as well as result in unreliable mail reception due to the dependency on multiple DNS lookups for every incoming SMTP connection.

Caution: Mail addressed to `postmaster` must **never** be rejected. Violation of this rule is sufficient cause for your domain to be disconnected from the Internet.

DNS_VERIFY is supplied as a sharable image on VMS (`PMDF_EXE:DNS_VERIFY.EXE`), as a sharable object library on UNIX (`/pmdf/lib/dns_verify`), and as a DLL on NT (`C:\pmdf\bin\dns_verify.dll`). It can be used from any of the mapping tables described in this chapter, using the routine callout described in Section 2.2.6.7, Customer-supplied Routine Substitutions, $[...].

On VMS, make sure that the PMDF_DNS_VERIFY logical name is set. For PMDF V6.2 and later, this logical is defined in PMDF_STARTUP.COM. If you are running PMDF V6.1 or earlier, add the following line to your PMDF_COM:PMDF_SITE_STARTUP.COM command procedure (create it if necessary):

```
$ DEFINE/SYSTEM/EXEC PMDF_DNS_VERIFY PMDF_EXE:DNS_VERIFY.EXE
```

DNS_VERIFY has 4 routines that can be called:

- `dns_verify`
- `dns_verify_domain`
- `dns_verify_domain_port`

- dns_verify_domain_warn

These are each described in the sections below.

Note that your mapping tables with DNS_VERIFY callouts can be tested by using the pmdf test -mapping utility (pmdf test/mapping on VMS).

16.1.8.1 dns_verify Routine

The dns_verify routine is the most general of the routines. It simply does a lookup in DNS of the domain name that you specify, which could be the domain from the SMTP MAIL FROM: command, or could be the domain name corresponding to the IP address in a blackhole list such as blackholes.mail-abuse.org. Any mapping table action can be taken if the domain exists or does not exist in DNS.

Note that when doing the DNS lookup, dns_verify looks for both an A DNS record and an MX DNS record.

The dns_verify routine performs the same type of action as the mailfromdnsverify channel keyword. Using DNS_VERIFY allows you more control over which connections trigger the DNS lookups.

The dns_verify routine's argument is four strings separated by " | ", as follows:

```
domainname[|success[|failure[|unknown]]]
```

- domainname is the name to look up in DNS.
- The success string is optional. If specified, it is the mapping table string to return if domainname is found in DNS. If not specified, the default is "$Y", to accept the message.
- The failure string is optional. If specified, it is the mapping table string to return if domainname is not found in DNS. If not specified, the default is "$N", to reject the message.
- The unknown string is optional. If specified, it is the mapping table string to return if there was a temporary DNS failure during lookup. If not specified, and the success string was specified, that string is used. If neither are specified, the default is "$Y".

Note that in the mapping table, the $'s need to be doubled. For example, to specify "$Y", you need to put in "$$Y".

An alternative separator can be used instead of " | ". To specify an alternative separator, specify it as the first character of the routine's argument. For example, to specify "+" as the separator, use the following syntax:

```
+domainname+success+failure+unknown
```

The success, failure, and unknown strings can contain the following format characters:

%a	If successful, the primary name for domainname. If there was no A record, but there was an MX record, this contains the domainname from the first MX record received from DNS.
%e	If not successful, the error message associated with the lookup.

%n If successful, the IP address found from the DNS lookup. If there was no A record, but there was an MX record, this contains the IP address corresponding to the first MX record received from DNS.

The following example shows a simple SEND_ACCESS mapping table entry on VMS to verify that the sender's hostname is in DNS:

```
SEND_ACCESS

  *tcp_*|*@*|*|*     \
$C$[pmdf_dns_verify,dns_verify,$3|$$Y|$$NInvalid$ host:$ $$3$-$ %e]$E
```

The following example shows a PORT_ACCESS mapping table entry on UNIX to check the IP address of the system sending the message:

```
PORT_ACCESS

  TCP|*|25|*|*      \
$C$[/pmdf/lib/dns_verify,dns_verify,\
$1|$$Y|$$N500$ Message$ refused$ from$ $$1$ -$ %e]$E
```

16.1.8.2 dns_verify_domain and dns_verify_domain_port Routines

The `dns_verify_domain` and `dns_verify_domain_port` routines are used to query the specified blackhole list and return pre-defined success, failure, and unknown messages. The same operation can be performed using the `dns_verify` routine, but with more complicated setup.

The `dns_verify_domain_port` routine is used in the PORT_ACCESS mapping table. The `dns_verify_domain` routine is used in the MAIL_ACCESS, SEND_ACCESS, and similar mapping tables.

The `dns_verify_domain` and `dns_verify_domain_port` routines perform the same type of action as the `DNS_VERIFY_DOMAIN` dispatcher option. Using DNS_VERIFY allows you more control over which connections trigger the DNS lookups.

The `dns_verify_domain` and `dns_verify_domain_port` routines' argument is three strings separated by ",", as follows:

```
ip-address,domainname[,text-string]
```

- `ip-address` is the IP address that you want to check against the blackhole list.
- `domainname` is the name of the blackhole list to check against, for example, `blackholes.mail-abuse.org`.
- `text-string` is optional. If specified, it is the text to return if no TXT record is available. If not specified, the default is "No Error Text Available".

`dns_verify_domain` and `dns_verify_domain_port` check the given list for the IP address. For example, if `ip-address` is 127.0.0.2, and `domainname` is `bl.spamcop.net`, `dns_verify_domain` and `dns_verify_domain_port` looks up the following name: 2.0.0.127.bl.spamcop.net. They first first look up the TXT record for that name, and if it is not available, they look up the A record.

The following examples show the use of these routines on VMS:

```
MAIL_ACCESS

  TCP|*|25|*|*|*|*|tcp_local|*|*|* \
$C$[pmdf_dns_verify,dns_verify_domain,$1,bl.spamcop.net]$E

PORT_ACCESS

  TCP|*|25|*|*     \
$C$[pmdf_dns_verify,dns_verify_domain_port,$1,bl.spamcop.net]$E
```

The approximate equivalent of the previous MAIL_ACCESS example using the dns_verify routine would be:

```
MAIL_ACCESS

  TCP|*|25|*.*.*.*|*|*|*|tcp_local|*|*|*    \
$C$[pmdf_dns_verify,dns_verify,+$4.$3.$2.$1.bl.spamcop.net+\
$$N$$X5.7.1|Blocked$ -$ see$ http://spamcop.net/bl.shtml?$$1.$$2.$$3.$$4+$$Y]$E
```

16.1.8.3 dns_verify_domain_warn Routine

The dns_verify_domain_warn routine performs the same DNS lookup as the dns_verify_domain and dns_verify_domain_port routines, but instead of rejecting the message if the DNS entry exists, it adds a new header line to the message. The dns_verify_domain_warn routine can be used in any of the ACCESS mapping tables.

The dns_verify_domain_warn routine's argument is four strings separated by ",", as follows:

```
ip-address,domainname[,text-string[,header]]
```

- ip-address, domainname, and text-string are the same as for dns_verify_domain and dns_verify_domain_port.
- header is optional. If specified, it is a string containing the header name, and other optional text, to include before the TXT record string or text-string value. The header name must be one that PMDF recognizes. The default is "X-Dispatcher: ".

The following example shows an ORIG_MAIL_ACCESS mapping table entry on NT to query spamcop.net:

```
ORIG_MAIL_ACCESS

 TCP|*|25|*|*|*|*|*|*|*|* \
$C$[C:\pmdf\bin\dns_verify.dll,dns_verify_domain_warn,$1,\
bl.spamcop.net,spamcop.net:$ entry$ found$ for$ $$1,\
X-Dispatcher:$ SPAMfilter$ (spamcop.net):$ ]$E
```

For a source IP address of 127.0.0.2, this example would return

```
$Y$AX-Dispatcher: SPAMfilter (spamcop.net): Blocked - see http://spamcop.net/bl.shtml?127.0.0.2
```

This is then added as a header to the message. To act upon this, create a system-wide, channel or user filter file containing a SIEVE command similar to:

```
if header :contains "X-Dispatcher" "SPAMfilter" { discard; }
```

16.1.9 SPF (Sender Policy Framework) and SRS (Sender Rewriting Scheme)

Note: SPF and SRS are not available for PMDF on Windows.

IMPORTANT: For OpenVMS sites running MultiNet, the ECO UCX_LIBRARY_EMULATION-090_A052 or later is required for SPF/SRS to function. For sites running TCPware, the ECO DRIVERS_V582P020 or later is required. For sites running TCP/IP Services, you need version 5.3 or later.

Sender Policy Framework (SPF) can be used to combat the problem of sender address forgery. It is a method of using special DNS records to say which systems are allowed to send mail for a given domain. PMDF's SPF implemention can be used to check the IP address of the system that PMDF is receiving a message from, against the domain name in the SMTP `MAIL FROM:` address to make sure the message is coming from an authorized sending system. The message can be accepted, rejected, or marked based on the results of the SPF checks.

Sender Rewriting Scheme (SRS) is an adjunct to SPF that is used by mail forwarders. SPF breaks mail forwarding because it rejects the forwarder as an unauthorized sender of the mail. If the forwarder uses SRS to rewrite the `MAIL FROM:` address, this problem is avoided.

For more information about SPF and SRS, see `http://www.openspf.org/`.

Using SPF/SRS is similar to DNS_VERIFY in that both are used to combat the problem of forged sender e-mail addresses. The same drawbacks and warnings apply (see Section 16.1.8).

As with DNS_VERIFY, SPF and SRS are supplied as a sharable images on OpenVMS (`PMDF_EXE:LIBSPFSHR.EXE` and `PMDF_EXE:LIBSRS2SHR.EXE`), and as sharable object libraries on Unix (`/pmdf/lib/libspf.so` and `/pmdf/lib/libsrs2.so`).

SPF can be called from the `FROM_ACCESS`, `ORIG_MAIL_ACCESS`, or `MAIL_ACCESS` mapping table. However, we recommend using the FROM_ACCESS mapping table for greatest efficiency since it is called only once, after the `MAIL FROM:` SMTP command instead of after every `RCPT TO:` SMTP command.

SRS is called from the `REVERSE` mapping table to encode the `MAIL FROM:` address when mail is forwarded. A different SRS routine is called from the `FORWARD` mapping table to decode `RCPT TO:` addresses when needed, as for example when an error response is sent back to an `MAIL FROM:` address that has been SRS-encoded. SPF and SRS are invoked using the routine callout described in Section 2.2.6.7, Customer-supplied Routine Substitutions, $[...].

Note that internal or trusted systems (such as mail gateways) should be excluded from SPF checks and SRS encoding by adding mapping table entries before the SPF or SRS entries in the mapping tables.

On Unix platforms, the symbols PMDF_SPF_LIBRARY and PMDF_SRS_LIBRARY are defined in /etc/pmdf_tailor to point to the SPF and SRS sharable object libraries, respectively. Similarly, on OpenVMS, the logical names PMDF_SPF_LIBRARY and PMDF_SRS_LIBRARY are defined in PMDF_STARTUP.COM to point to the sharable images. We also recommend that you install the SPF and SRS sharable images for efficiency. The following commands to do this may be put in your PMDF_COM:PMDF_SITE_STARTUP.COM file:

```
$ INSTALL ADD PMDF_SPF_LIBRARY/OPEN/HEAD/SHARE
$ INSTALL ADD PMDF_SRS_LIBRARY/OPEN/HEAD/SHARE
```

16.1.9.1 Configuring SPF

LIBSPFSHR has three routines that can be called:

- spf_lookup
- spf_lookup_reject_fail
- spf_lookup_reject_softfail

These are each described in the sections below.

Note that your mapping tables with SPF callouts can be tested by using the pmdf test -mapping utility (pmdf test/mapping on OpenVMS).

16.1.9.1.1 spf_lookup Routine

This routine does the SPF lookup and adds a Received-SPF: header upon successful completion, or a header of your choice upon error (such as if there are no SPF records).

The spf_lookup routine has four arguments, separated by commas. The callout would look like this:

```
$[PMDF_SPF_LIBRARY,spf_lookup,sending-ip,from-address,domain-name,header]$
```

The parameters are as follows:

- sending-ip is the sending IP address
- from-address is the MAIL FROM: address
- domain-name is the domain name to display in the Received-SPF: headers; this is expected to be the name of the local system that is doing the SPF lookup
- header is the type of header to add upon error

The following example adds a Received-SPF: header, or in the case where there is no SPF record, a X-PMDF-SPF: header:

```
FROM_ACCESS

  TCP|*|25|*|*|SMTP*|*|tcp_*|*|*  $C$[PMDF_SPF_LIBRARY,\
spf_lookup,$1,$6,example.com,X-PMDF-SPF]$E
```

Example Received-SPF: and X-PMDF-SPF: headers would be:

```
Received-SPF: pass (example.com: domain of remotesys.com designates 10.0.0.1
 as permitted sender) client-ip=192.168.0.1; envelope-from=joe@remotesys.com;
X-PMDF-SPF: (recv=mail.example.com, send-ip=10.0.0.2) Could not find a
 valid SPF record
```

16.1.9.1.2 spf_lookup_reject_fail and spf_lookup_reject_softfail Routines

Both of these routines do the SPF lookup and then reject the message if it gets a *fail* result. The `spf_lookup_reject_softfail` also rejects the message if it gets a *softfail* result. They both add a `Received-SPF:` header upon successful completion.

These routines have four arguments, separated by commas. The callouts would look like this (for example on Unix):

```
$[/pmdf/lib/libspf.so,spf_lookup_reject_fail,sending-ip,from-address,domain-name,header]$
$[/pmdf/lib/libspf.so,spf_lookup_reject_softfail,sending-ip,from-address,domain-name,text]$
```

The parameters are as follows:

- `sending-ip` is the sending IP address
- `from-address` is the `MAIL FROM:` address
- `domain-name` is the domain name to display in the `Received-SPF:` headers; this is expected to be the name of the local system that is doing the SPF lookup
- `text` is the rejection text to use

The following example rejects mail that fails the SPF lookup:

```
FROM_ACCESS

  TCP|*|25|*|*|SMTP*|*|tcp_*|*|*  $C$[/pmdf/lib/libspf.so,\
spf_lookup_reject_softfail,$1,$6,example.com,Rejected$ by$ SPF$ lookup]$E
```

16.1.9.2 Configuring SRS

To configure PMDF to use SRS, changes are required to the PMDF option file `pmdf_table:option.dat`, configuration file `pmdf_table:pmdf.cnf`, and mapping file `pmdf_table:mappings`.

16.1.9.2.1 Option File Changes

Two options need to be added to the PMDF option file: `REVERSE_ENVELOPE`, and `USE_REVERSE_DATABASE`.

The `REVERSE_ENVELOPE` option needs to be set to 1 to tell PMDF to apply the REVERSE mapping table to `envelope from` addresses (by default the REVERSE mapping table is only applied to header addresses). The `USE_REVERSE_DATABASE` option should be set to a value of 266. This value turns on address reversal processing (the REVERSE mapping table in this case), as well as specifying that the destination channel be prepended to the address when probing the REVERSE mapping table.

```
REVERSE_ENVELOPE=1
USE_REVERSE_DATABASE=266
```

16.1.9.2.2 Configuration File Changes

By default, the REVERSE mapping table is checked for every destination channel. For SRS, normally you'd only want to encode the `envelope from` address for messages that are being forwarded to the internet, for example, being sent out the `tcp_local` channel.

To avoid unnecessary processing, the checking of the REVERSE mapping table can be restricted to when `tcp_local` is the destination channel. This is done by using the `noreverse` and `reverse` channel keywords in the `pmdf.cnf` file.

Place the `noreverse` channel keyword on the `defaults` line in `pmdf.cnf` to turn off checking of the REVERSE mapping table by default. To turn the checking on for the `tcp_local` channel, add the `reverse` channel keyword to the `tcp_local` channel definition. If there are any other channels you use that you wish to use SRS on, put the `reverse` channel keyword on those channel definitions as well.

16.1.9.2.3 Mapping File Changes

The SRS library has two routines that can be called:

- `pmdf_srs_forward`
- `pmdf_srs_reverse`

The `pmdf_srs_forward` routine is called from the REVERSE mapping table to encode the `envelope from` address. The `pmdf_srs_reverse` routine is called from the FORWARD mapping table to decode any SRS-encoded `envelope to` addresses. These routines are further described in the sections below.

Note: Notice that the name of the routine is opposite from the name of the mapping table that it is used in.

Note that your mapping tables with SRS callouts can be tested by using the `pmdf test -mapping` utility (`pmdf test/mapping` on OpenVMS).

16.1.9.2.3.1 pmdf_srs_forward Routine And The REVERSE Mapping Table

The `pmdf_srs_forward` routine takes an address and encodes it as defined by the SRS rules. It returns the SRS-encoded address.

This routine has three arguments, separated by commas. The callout would look like this:

```
$[PMDF_SRS_LIBRARY,pmdf_srs_forward,from-address,secret,domain-name]$
```

The parameters are as follows:

1. `from-address` is the `MAIL FROM:` address to encode.
2. `secret` is a secret word you must specify for use by the encoding process.
3. `domain-name` is the local domain to use.

When adding the pmdf_srs_foward callout to the REVERSE mapping table, you must specify the $:E flag on the entry. The option.dat option REVERSE_ENVELOPE (as described above) causes PMDF to apply the REVERSE mapping table to the envelope from address as well as to header addresses. However, SRS should be applied **only** to the envelope address and not to header addresses. Specifying the $:E flag on your REVERSE mapping table entry achieves this.

For best efficiency, SRS should only be applied to messages going out via the tcp_local channel (or other external channel). The option.dat option USE_REVERSE_DATABASE (as described above) specified with a value of 266 causes PMDF to prepend the destination channel to the address when the REVERSE mapping table is probed. To apply the mapping table entry to only tcp_local, specify "tcp_local | " at the front of your reverse mapping table entries.

Note: Note that if you are already using the REVERSE mapping table for something else, this setting of USE_REVERSE_DATABASE causes the destination channel to be prepended for **all** REVERSE mapping table entries.

Also, SRS should only be applied to messages that are being forwarded, and not to locally-generated messages. To do this, add additional entries to the REVERSE mapping table that exempts messages that have a local envelope from address.

The following example encodes mail being forwarded by example.com:

```
REVERSE

 tcp_local|*@*example.com  $N
 tcp_local|*@*             $:E$[PMDF_SRS_LIBRARY,\
pmdf_srs_forward,$0@$1,mysecret,example.com]$E
```

16.1.9.2.3.2 pmdf_srs_reverse Routine And The FORWARD Mapping Table

The pmdf_srs_reverse routine takes an SRS-encoded address and decodes it back into the original address.

This routine has three arguments, separated by commas. The callout would look like this:

```
$[PMDF_SRS_LIBRARY,pmdf_srs_reverse,to-address,secret,domain-name]$
```

The parameters are as follows:

1. to-address the SRS-encoded RCPT TO: address to decode.
2. secret is the secret word that was used during the encoding process.
3. domain-name is the local domain that was used during the encoding process.

Adding a callout to the pmdf_srs_reverse routine from the FORWARD mapping table is required to decode any SRS-encoded envelope to addresses. For example, to handle responses to mail that you sent with an SRS-encoded envelope from address. Note that when you add the callout, you must also specify the $D flag on the entry, to tell PMDF to send the resulting address back through the rewrite process. This is so the mail can be forwarded on successfully to the real original sender.

Since SRS decoding should only be applied to SRS-encoded addresses, you can select for this by checking for addresses that have two equal signs in the username part.

The following example decodes SRS-encoded addresses received by example.com:

```
FORWARD

  *=*=*@example.com    $[PMDF_SRS_LIBRARY,\
pmdf_srs_reverse,$0=$1=$2@example.com,mysecret,example.com]$D
```

16.1.9.2.4 The Secret Word

The secret word specified in the mapping table callouts is used by the SRS encoding and decoding to make sure that the SRS-encoded address is not forged. You do not have to change your secret word at all, but if you wish to do so, the decoding routine needs to know all of the secret words ever used so that it can properly decode messages that were encoded using a previous secret word.

The SRS decoding routine will check for the file `pmdf_table:srs_secret.dat` for a list of previous secret words. Whenever you change your secret word, add the old secret word to this file (one word per line).

16.2 Mailbox Filters

Individual users can use personal PMDF mailbox filters to prevent delivery of unwanted mail messages to their mailboxes, or to define vacation notices. Or the PMDF manager can create channel level filters or a system wide filter.

Mailbox filters are text files containing commands in the SIEVE language (see Section 16.2.7). Mailbox filter files may be created by hand using any text editor. PMDF also supplies a web-based interface for creating mailbox filters, see Section 16.2.6. System managers can customize this web interface as described in 16.2.6.

Through the web-based interface, users can construct and manage the screening rules applied by their personal mailbox filters and configure their vacation notices, or the PMDF manager can construct an manage the screening rules for channel level filters or a system wide filter. Personal mailbox filters are supported for use with the l (lowercase “L”), msgstore, and popstore channels. Channel level filters are supported for all channels.

The web-based interface allows you to set up eight distinct filters: four to identify messages to always keep, the "Accept filters"; four to identify messages to always throw away, the "Discard filters". The `Accept` and `Discard` filters operate on envelope and header source addresses, header destination addresses, and phrases or words appearing in the `Subject:` header line or body of the message. The eight filters are thus known by the names `Accept From`, `Accept To`, `Accept Subject`, `Accept Body`, `Discard From`, `Discard To`, `Discard Subject`, and `Discard Body`.

The web-based interface also allows users to set up vacation notices. See Section 16.2.8. Note that if you set up a vacation notice for a channel level filter or the system filter, it will just be ignored.

Users’ personal mailbox filters are applied first. If a personal mailbox filter explicitly accepts or rejects a message, then filter processing for that message copy finishes. But if the recipient user had no mailbox filter—or if the user’s mailbox filter did not explicitly apply to the message in question—PMDF next applies the channel level filter. If the

channel level filter explicitly accepts or rejects a message, then filter processing for that message finishes. Otherwise, PMDF next applies the system filter file, if there is one.

By default, each user has no mailbox filter. When a user uses the web-based interface, and mailbox filtering is enabled for the channel which delivers mail to their mailbox, a mailbox filter is created for them. Each mailbox filter is stored in a disk file, the location of which is controlled with the `filter` channel keyword as described in Section 16.2.1.

Channel filters are not present by default. If channel level filtering is enabled for a channel, then the PMDF manager can create a channel level filter using the web-based interface, by authenticating as the "address" `@`*channel-host-name* where *channel-host-name* is the official host name of the channel in question, and providing the password for the PMDF server account.

The PMDF manager can create a system level filter using the web-based interface by authenticating as the "address" `@`—that is, specifying the single character `@` as the "address"—and providing the password for the PMDF server account.

16.2.1 The `filter` Channel Keyword

The `filter` channel keyword enables message filtering on the channels to which it is applied. The keyword has one required parameter which specifies the path to the filter files associated with each envelope recipient who receives mail via the channel.

The syntax for the `filter` channel keyword is

```
filter URL-pattern
```

where *URL-pattern* is a URL which, after processing special substitution sequences, yields the path to the filter file for a given recipient address.

URL-pattern can contain special substitution sequences which, when encountered, are replaced with strings derived from the recipient address,

local-part`@`*host.domain*

in question. These substitution sequences are given in Table 16–2.

Table 16–2 `filter` Channel Keyword Substitution Strings

Sequence	Substitution string
`$$`	Substitute in the $ character
`$A`,`$a`	Substitute in the address, *local-part*`@`*host.domain*
`$D`,`$d`	Substitute in *host.domain*
`$H`,`$h`	Substitute in *host*
`$L`,`$l`	Substitute in *local-part*
`$S`,`$s`	Substitute in subaddress or folder name. Used for `fileinto` channel keyword only.

Table 16–2 (Cont.) `filter` Channel Keyword Substitution Strings

Sequence	Substitution string
`$U,$u`	Substitute in *local-part* less any underscore or tilde prefixes and less any subaddress postfix
`$~`	Substitute in the file path for the home directory associated with the local part of the address

The `filter` channel keyword can be used to specify filters for the l (lowercase "L"), msgstore, and popstore channels. Suggested usages of the `filter` channel keyword are given in Section 16.2.1.1 and Section 16.2.1.2.

16.2.1.1 Keyword Usage with the Local Channel

For the local channel, the suggested usage on OpenVMS systems is

```
filter file:///$~PMDF_MAILBOX_FILTER.
```

This will place the mailbox filter file in each user's login directory. The name of the filter file will be `PMDF_MAILBOX_FILTER.;` and the file will be owned by the user.

On UNIX platforms, the suggested usage is

```
filter file:///pmdf/user/l/$u.filter
```

This places the users' mailbox filter files in the PMDF user profile area, `/pmdf/user/`, under the `l` (lowercase L for "l"ocal channel) subdirectory. The mailbox filter files will be owed by the user pmdf.

Note: Make sure that the `/pmdf/user/l` directory is owned by the user `pmdf`.

16.2.1.2 Keyword Usage with the msgstore and popstore Channels

For a msgstore or popstore channel, use

```
filter popstore:$U
```

No other usage with msgstore or popstore channels is supported. The PMDF MessageStore and PMDF popstore will place each mailbox filter file in the same directory as the associated user's PMDF profile file. When a PMDF MessageStore or PMDF popstore user account is deleted, any associated mailbox filter file will be deleted. Likewise, when a PMDF MessageStore or PMDF popstore account is renamed, any associated filter file will be renamed.

The PMDF MessageStore considers subaddresses (see Section 3.1.1.6) to indicate folders for delivery purposes. For msgstore channels that have mailbox filtering enabled, use of the Sieve `fileinto` operator can be enabled via use of the `fileinto` channel keyword on the channel. The usual usage is:

```
fileinto $U+$S@$D
```

so that a msgstore channel with mailbox filtering with `fileinto` enabled would look something like:

```
msgstore filter popstore:$U fileinto $U+$S@$D
messagestore-domain-name
```

16.2.2 Channel Level Filter Files

The `destinationfilter` channel keyword (synonymous with the obsolete `channelfilter` keyword) enables message filtering on the destination channels to which it is applied. The `sourcefilter` channel keyword enables message filtering on the source channels to which it is applied. These keywords each have one required parameter which specifies the path to the corresponding channel filter file associated with the channel.

The syntax for the `destinationfilter` channel keyword is

```
destinationfilter URL-pattern
```

and similarly the syntax for the `sourcefilter` channel keyword is

```
sourcefilter URL-pattern
```

where *URL-pattern* is a URL specifying the path to the filter file for the channel in question. For instance, suggested usage on OpenVMS is

```
destinationfilter file:///PMDF_TABLE:channel-name.filter
```

or on UNIX or NT

```
destinationfilter file:///pmdf/table/channel-name.filter
```

where *channel-name* is the name of the channel, *e.g.*, l, tcp_local, ln_local, *etc.*

Channel filter files must be world-readable. Note that a channel filter file will be owned by the PMDF server account—usually PMDF on OpenVMS, or `pmdf` on UNIX.

Important Note: The web-based interface (see Section 16.2.6 creates only destination filters (`destinationfilter` keyword). You cannot use the web interface to create source filters (`sourcefilter` keyword).

16.2.3 The System Wide Filter File

On OpenVMS, the system wide filter file is

```
PMDF_TABLE:pmdf.filter
```

On UNIX, the system wide filter file is

```
/pmdf/table/pmdf.filter
```

On NT, the system wide filter file is

```
C:\pmdf\table\pmdf.filter
```

The system wide filter file must be world readable. It is used automatically, if it exists.

When using a compiled configuration, the system wide filter file is incorporated into the compiled configuration.

16.2.4 Mailbox Filter Authentication

When a user creates or modifies their personal mailbox filter, the user must authenticate himself.

When the PMDF manager creates or modifies a channel level filter or the system wide filter, the manager must authenticate with the PMDF server account[4] password and as the "address" `@`*`channel-host-name`* where *`channel-host-name`* is the official host name of the channel in question for a channel level filter, or as the "address" `@` for the system filter file. For instance, on a system with official local host name example.com, the PMDF manager would authenticate using the "address" `@example.com`—the `@` character followed by the host name *with no username*—for a channel level filter, or would authenticate using the "address" `@`—*the at sign character alone, with no username or host name*—for a system level filter.

The PMDF security configuration controls just what source of authentication material this authentication will be performed against, *e.g.*, PMDF user profile password (PMDF popstore or PMDF MessageStore user profile), PMDF password database password, system password file password, *etc.*; see Section 14.2. For mailbox filter connections handled by the DEFAULT security rule set of PMDF's implicit security configuration, authentication will be performed preferentially against the PMDF user profile, if the user has a PMDF user profile entry, if not then against the PMDF password database, if the user has an entry in it, and finally, only if the user has neither sort of entry, against the system password file.

In the particular case when authentication is performed against the PMDF password database, note that PMDF will check just which channel a user matches in order to decide which of the user's (possible multiple) PMDF password database entries to compare against. For a user matching a msgstore channel, the mailbox filter query will preferentially use the user's /SERVICE=IMAP entry, but if such an entry does not exist will fall through to the user's /SERVICE=DEFAULT entry. For a user matching a popstore channel, the mailbox filter query will preferentially use the user's POP service-specific entry in the password database, but if such an entry does not exist will fall through to the user's DEFAULT entry in the password database. For a user matching the local channel, the mailbox filter query will use the user's DEFAULT entry. See Section 14.7 for an additional discussion of the PMDF password database.

[4] The PMDF server account is usually PMDF on OpenVMS, or `pmdf` on UNIX; on NT use `Administrator`.

16.2.5 Routing Discarded Messages Out the FILTER_DISCARD Channel

By default, messages discarded via a mailbox filter are immediately discarded (deleted) from the system. However, when users are first setting up mailbox filters (and perhaps making mistakes), or for debugging purposes, it can be useful to have the deletion operation delayed for a period.

To have mailbox filter discarded messages temporarily retained on the PMDF system for later deletion, first add a FILTER_DISCARD channel to your PMDF configuration, *e.g.*:

```
filter_discard notices 7
FILTER-DISCARD
```

with the `notices` channel keyword specifying the length of time (normally number of days) to retain the messages before deleting them. Then set the option FILTER_DISCARD=2 or FILTER_DISCARD=3 in the PMDF option file; see Section 7.3.3.

Messages in the FILTER_DISCARD queue area should be considered to be in an extension of users' personal wastebasket folders. As such, note that warning messages are never sent for messages in the FILTER_DISCARD queue area, nor are such messages returned to their senders when a bounce or return is requested. Rather, the only action taken for such messages is to eventually silently delete them, either when the final `notices` value expires, or if a manual bounce is requested using a utility such as `pmdf return`.

16.2.6 Web Interface

The following section documents the web-based interface for maintaining per-user, per-channel, or system-wide mailbox filters. Use of this facility requires both TCP/IP support and a web client. Moreover, the PMDF HTTP server must be configured to serve out this interface; see Section 16.2.6.1 for details.

The web interface is intended for use by individual users who have their mail delivered to them via one of the l (lowercase "L"), msgstore, or popstore channels, or by the PMDF manager for managing channel level filters or the system filter file. When users attempt to use the interface, they must supply their e-mail address and their password in order to access their mailbox filter. Typically, a l (lowercase "L") channel user will provide their login password, a msgstore channel user will provide their IMAP password, while a user whose mail is delivered by a popstore channel would provide their POP password. This behavior is controlled through the PMDF authentication services interface as described in Chapter 14. The PMDF manager must supply the "address" `@`*`channel-host-name`* for a channel level filter, where *`channel-host-name`* is the official host name of the channel in question, or the single character `@` for the system filter file, and the password for the PMDF server account.

Note: For channels, the web interface creates only destination filters (see Section 16.2.2, `destinationfilter` keyword). You cannot use the web interface to create channel level source filters (`sourcefilter` keyword).

To connect to the interface with your web browser, open the URL

```
http://host:7633/mailbox_filters/
```

In place of *host*, use the actual IP host name of the system running the PMDF HTTP server. If you chose to run the PMDF HTTP server on a port other than port 7633, then specify that port number in place of `7633` in the above URL.

Once connected to the introductory web page, links to help and various mailbox filtering activities can be followed. A sample of the introductory web page and a few of the other default, Process Software supplied web pages can be found in the appropriate edition of the *PMDF User's Guide*.

16.2.6.1 Configuring the HTTP Server to Serve Out the Web Interface

If you have not already configured the PMDF Dispatcher, then do so now. See the Chapter on configuring the PMDF Dispatcher in the appropriate edition of the *PMDF Installation Guide* for directions on configuring the Dispatcher. As part of configuring the Dispatcher, the HTTP server is configured.

Configuring the HTTP server to serve out the web based interface is simply a matter of ensuring that the HTTP server's configuration file contains a proper mailbox filter path definition. The current version of the Dispatcher configuration utility generates an appropriate definition automatically; but if you have an older configuration generated in PMDF V5.1, you can need to add the definition manually and then restart the HTTP server. The `http.cnf` file in the PMDF table directory needs to include the lines

```
[PATH=/mailbox_filters/]
GET=PMDF_MAILBOX_FILTERS_CGI
POST=PMDF_MAILBOX_FILTERS_CGI
```

If these lines were not previously present, after adding them restart the HTTP server with the OpenVMS command

```
$ PMDF RESTART HTTP
```

or the UNIX command

```
# pmdf restart http
```

Or on NT restart the entire Dispatcher with the command

```
C:\> pmdf restart dispatcher
```

16.2.6.2 The Mailbox Filters Option File

An option file can be used to adjust various options for the web interface. On OpenVMS, the option file is

```
PMDF_TABLE:mailbox_filters_option.
```

On UNIX systems, it is the file

```
/pmdf/table/mailbox_filters_option
```

On NT systems, it is typically the file

```
C:\pmdf\table\mailbox_filters_option
```

Available options are:

AUTHENTICATION_ERROR (file-name)
Name of a formatting file in the `pmdf_root:[www.mailbox_filters]` (OpenVMS) or `/pmdf/www/mailbox_filters/` (UNIX) or `C:\pmdf\www\mailbox_filters\` (NT) directory to use for authentication errors. The default is

```
AUTHENTICATION_ERROR=auth_error.txt
```

DEFAULT_HOST (host-name)
Name of a host to append to a supplied username which lacks an "@". By default, the local host name is used.

GENERAL_ERROR (file-name)
Name of a formatting file in the `pmdf_root:[www.mailbox_filters]` (OpenVMS) or `/pmdf/www/mailbox_filters/` (UNIX) or `C:\pmdf\www\mailbox_filters\` directory to use for general errors which occur when either (1) no error_format file can be discerned from the HTTP command, or (2) when the HTTP command has not yet been processed. The default is

```
GENERAL_ERROR=error.txt
```

16.2.7 SIEVE

PMDF mailbox filters implement the SIEVE RFC (3028), plus the vacation draft (DRAFT-SHOWALTER-SIEVE-VACATION-04.TXT), and a couple of other extensions.

RFC 3028 and the vacation draft are available in the PMDF RFC directory, if you chose to install them. The PMDF RFC directory is `PMDF_ROOT:[DOC.RFC]` on VMS, `/pmdf/doc/rfc` on UNIX, and `C:\pmdf\doc\rfc` on NT.

16.2.7.1 Standard SIEVE Commands

The `SIEVE` commands are standardized in RFC 3028. This section gives a summary of the `SIEVE` commands, including any related PMDF-specific information. Refer to RFC 3028 for more information about the `SIEVE` commands.

Note that for list types, such as the list of strings to search for in a header, there is a limit on the number of entries that can be in a single list. This limit is specified by the `MAX_LIST_SIZE` option (see Section 7.3.3).

16.2.7.1.1 Comments

PMDF supports the two comment delimiters defined in RFC 3028.

The first kind are hash comments, these begin with the hash sign (#). These comments usually start at the beginning of the line. They always cause the rest of the line to be considered a comment.

The second kind are bracketed comments. They begin with the two-character string "/*" and end with the two-character string "*/".

16.2.7.1.2 Control structures

if <test1: test> <block1: block>
elsif <test2: test> <block2: block>
else <block3: block>
Standard 'if' control structure similar to other languages.

require <capabilities: string-list>
Some commands in the filter file are optional or are extensions to RFC 3028. If any of these are used, they must be specified in a `require` command at the beginning of the file. The capabilities that PMDF supports are: `envelope`, `fileinto`, `reject`, `vacation`.

stop
Ends all processing of the filter file.

16.2.7.1.3 Common arguments

ADDRESS-PART: ":localpart" / ":domain" / ":all"
`ADDRESS-PART` is used in comparisons against addresses to indicate which parts of the address to look at. `:localpart` refers to the left side of the `@`, `:domain` refers to the right side of the `@`, and `:all` refers to the entire address. The default is `:all`.

COMPARATOR: ":comparator" <comparator-name: string>
`COMPARATOR` indicates what kind of character comparison to do. The possible values for `comparator-name` are: `i;octet` for case-sensitive matching, and `i;ascii-casemap` for case-insensitive matching. The default is `i;ascii-casemap`.

MATCH-TYPE: ":is" / ":contains" / ":matches"
`MATCH-TYPE` indicates what kind of comparison to do for tests that compare one string to another. The `:is` match-type requires an exact string match. The `:contains` match-type requires a substring match. The `:matches` match-type is for wildcard matching. Using `:matches` enables the wildcard characters "*" (0 or more characters) and "?" (single character). The default is `:is`.

16.2.7.1.4 Test commands

address [ADDRESS-PART] [COMPARATOR] [MATCH-TYPE] <header-names: string-list> <key-list: string-list>
The `address` test returns `true` if any header in the `header-names` list contains any key in the `key-list` in the specified part of the address, doing the comparison as indicated by the comparator and match keywords. Similar to the `header` test, but only applies to headers with addresses as values.

allof <tests: test-list>
The `allof` test performs a logical AND on the tests supplied to it.

anyof <tests: test-list>
The anyof test performs a logical OR on the tests supplied to it.

envelope [ADDRESS-PART] [COMPARATOR] [MATCH-TYPE] <envelope-part: string-list> <key-list: string-list>
The envelope test returns true if any of the envelope headers in the envelope-part list contains any key in the key-list in the specified part of the address, doing the comparison as indicated by the comparator and match keywords. The possible values for envelope-part are "to" and "from".

exists <header-names: string-list>
The exists test returns true if all of the headers in the header-names list exist within the message.

false
The false test always evaluates to false.

header [COMPARATOR] [MATCH-TYPE] <header-names: string-list> <key-list: string-list>
The header test returns true if any header in the header-names list contains any key in the key-list, doing the comparison as indicated by the comparator and match keywords.

not <test>
Evaluates the specified test, and reverses the result.

size <":over" / ":under"> <limit: number>
The size test compares the size of the message to the specified limit. If :over is specified, it returns true if the size is larger than the limit. If :under is specified, it returns true if the size is smaller than the limit. It never returns true if the size is equal to the limit.

true
The true test always evaluates to true.

16.2.7.1.5 Action commands

discard
Throws the message away. If the FILTER_DISCARD option (see Section 7.3.3) is set to 2 or 3, the message is placed on the filter_discard channel, othewise it is immediately deleted by the bitbucket channel.

fileinto <folder: string>
Supported for MessageStore only. Delivers the message to the specified folder instead of the INBOX. The maximum number of fileinto commands allowed is specified by the MAX_FILEINTOS option (see Section 7.3.3). The fileinto channel keyword must be specified on the channel (see Section 16.2.1.2).

keep
This is the default action. Delivers the message to the INBOX.

redirect <address-list: string-list>
Forwards the message to all of the addresses listed in address-list. Put each address in double quotes and separate them with commas. The maximum number of redirect commands allowed is specified by the MAX_FORWARDS option (see Section 7.3.3).

The addresses in address-list may contain any of the substitution strings listed in Table 16–2.

reject <reason: string>
Rejects the message and sends a rejection notice back to the sender containing the reason for the rejection.

16.2.7.2 The SIEVE Vacation Command

The SIEVE vacation command is described in DRAFT-SHOWALTER-SIEVE-VACATION-04.TXT and summarized here.

The syntax for the vacation command is:

```
require "vacation";
vacation [":days" number] [":addresses" string-list] [":subject" string]
         [":mime"] <reason: string>;
```

The optional :days parameter is used to specify the period in which addresses are kept and are not responded to. The valid range is 1 to 30 days. The default is 7.

Only messages that are addressed directly to the user are responded to with a vacation notice. The optional :addresses parameter is used to allow the user to specify other addresses that PMDF should consider to belong to the user.

The optional :subject parameter is used to specify the subject of the vacation notice message. If it is not specified, the subject is Re: followed by the original message's subject.

The optional :mime parameter is for advanced users only. It tells PMDF that the reason argument text is formatted as a valid MIME part, including the MIME headers.

The reason argument is the text of the vacation notice. Formatting using multiple lines, blank lines, and tab characters is supported.

The vacation command is only supported in users' personal filter files. For more information on vacation notices, see Section 16.2.8.

16.2.7.3 PMDF SIEVE Extensions

PMDF also implements the following commands, which are not standard. They are listed here using the format used in RFC 3028, and using keywords (such as COMPARATOR) defined in RFC 3028.

body [:contains] [COMPARATOR] <key-list: string-list>
The body test searches the body of the message for strings in key-list. The body test is case-insensitive by default. COMPARATOR can be specified to make it case-sensitive.

debug <message: string>
The debug command prints out the message in the currently active debug log file.

hold
The hold command causes PMDF to hold the message as a .HELD file. The hold command is allowed in the system filter file and channel level filters only.

16.2.7.4 Example Filter File

The following is a simple example of a filter file:

```
require ["fileinto","vacation"];
if body :contains "free stuff"
    { discard; }
elsif address :localpart "to" "info-pmdf"
    { fileinto "info-pmdf"; }
else
    { vacation :days 5 :subject "I'm on vacation" "see you next week"; }
```

16.2.8 Vacation Notices

Users can establish a vacation notice using the `vacation` SIEVE command (see Section 16.2.7.2) or using the web-based interface (see Section 16.2.6). A vacation notice sends an automatic reply to mail messages received by the user. See the appropriate edition of the *PMDF User's Guide* for more information.

Note that the `vacation` SIEVE command is only supported for users' personal mailbox filters. If it appears in a channel filter or system filter file, it will not cause a syntax error, but no vacation notices will be sent out.

Vacation notices are always sent to the envelope `From:` address. Vacation notices are never sent to mailing lists or other addresses which are most likely from automated mailers (see vacation exceptions option file, Section 16.2.8.1).

The vacation notice itself is formatted as a `multipart/report` message, where the report type is `delivery-status`. The text of the notice (the `reason` argument to the `vacation` command) is included in the first part of the multipart message.

PMDF keeps track of all of the addresses which have been responded to, per vacation message, in the vacation notice history file. This file is in the same directory as the mailbox filter file. Its name is the name of the filter file with `-vnhf` appended to it. This is a text file, but it should not be edited by users or system managers.

Note: For OpenVMS systems, make sure that the directories containing the mailbox filter files (typically the users' home directories) do NOT have a version limit of 1. The vacation notice history file requires the ability to have at least 2 versions.

16.2.8.1 Vacation Exceptions Option File

As required by the SIEVE vacation command draft document (DRAFT-SHOWALTER-SIEVE-VACATION-04.TXT), PMDF has a list of addresses that vacation notices are not sent to (because they are most likely from automated mailers). PMDF also has a list of headers that indicate a mailing list, since vacation notices must not be sent to mailing lists. These lists of addresses and headers are specified in the vacation exceptions option file, `vacation_exceptions.opt`, located in the PMDF table directory. PMDF supplies a default list during installation. System managers may edit this file to add new headers and addresses, or remove ones already there. Note that this file is replaced upon installation (except on VMS), so any changes must be saved by the system manager so they are not lost.

The vacation exceptions file contains a list of headers and optional values. Note that only headers recognized by PMDF are supported in the file; all others are ignored. Any of the header lines described in this manual may be specified, plus any of the header lines standardized in RFC 822, RFC 987, RFC 1049, RFC 1421, RFC 1422, RFC 1423, RFC 1424, RFC 2156, and RFC 2045.

When PMDF is about to send a vacation notice in response to a mail message, it searches the mail message first for each header in the file. If any of the headers is present with the specified value, the vacation notice is not sent.

A header can be included with no value (make sure to include the colon ":" after the header name), indicating that the header's presence alone is sufficient to suppress the vacation notice. Wildcards are supported in the value part ("*" and "?"), but not in the header name. Use the backslash character ("\") to quote the wildcards.

16.2.9 Checking Your Changes

Important: After you have made changes to any mailbox filter file, it is important to verify that it is working correctly, especially if it was edited manually. If the filter file is not working, for example if it has a syntax error, mail delivery could be interrupted.

The syntax of filter files can be partially verified using the following command:

```
# pmdf test -rewrite -filter mailbox (UNIX and NT)
```

or

```
$ pmdf test/rewrite/filter mailbox  (VMS)
```

This command only checks some elements of the filter file. The best way to check a filter file is to send mail through it.

Note that the web-based interface always generates syntactically valid filter files.

17 The UNIX Local Channel

The local channel is used to deliver messages to addresses on the local host. On UNIX, the local channel is the `l` (lowercase *L*) channel.

When using a mail user agent on the local system to send mail (to anywhere), `/pmdf/bin/sendmail` is invoked as the replacement for `sendmail` to queue the messages to the appropriate queues, and then the channel programs for those queues will process the messages. Such local mail user agents can include `mail`, `mailx`, `mailtool`, `Pine`, *etc*.—any mail user agent that normally invokes sendmail.

The `pmdf configuration` utility always generates a local channel.

17.1 /pmdf/bin/sendmail

`/pmdf/bin/sendmail` is designed for compatibility with mail user agent applications. However, it is not intended to be 100% compatible with `sendmail` when it comes to command line options for use by human users in submitting messages or management activities. PMDF's design has some fundamental differences from `sendmail`'s design; PMDF as a whole provides a safer implementation of all `sendmail`'s functionality, plus a great many additional capabilities. Not all `sendmail` command line options make sense for PMDF's sendmail replacement; the underlying functionality can be provided by PMDF in a way other than `sendmail` options.

Instead, for submitting messages from the command shell, see the `pmdf send` utility, described in the UNIX Edition of the *PMDF User's Guide*. For checking on the PMDF queues, see the `pmdf qm` utility described in Section 30.2.2. For listing currently executing PMDF processes, see the `pmdf process` utility described in Chapter 30. Or for other management activities such as testing address rewriting, see additional PMDF utilities described in Chapter 30.

The command line options recognized by PMDF's `sendmail` replacement are shown in Table 17–1.

Table 17–1 `/pmdf/bin/sendmail` **options**

Option	Usage
`-help`	Help; gives list of options supported
`-h`	Synonym for `-help`
`-t`	Use headers for envelope addresses
`-odq`	Enqueue only, with priority set to nonurgent
`-oee`	No error status returned to shell

Table 17–1 (Cont.) `/pmdf/bin/sendmail` **options**

Option	Usage
`-f`*user*	Set `From:` username to `user`. In general, must be `root`, or in the `pmdf_world` group, to set user to other than one's own username, though users in the `pmdf_world_`*username* group are also allowed to specify `-f=`*username*.
`-bs`	SMTP dialog to `stdin/stdout` — all other options ignored
`-oi`	Line containing single dot (.) does not terminate message
`-i`	Synonym for `-oi`
`-r`*user*	Synonym for `-f`*user*
`--`	Any command line arguments after this option are ignored
The following are always set when sending a message	
`-oem`	Mail error message back to user
`-om`	The sender address will not be stricken, if present, from the recipient list during alias expansion
`-m`	Synonym for `-om`
`-odb`	Background delivery (asynchronous)
The following are ignored without error messages	
`-v`	Set verbose mode
`-oo`	Old header format (with spaces instead of commas between addresses)
`-odi`	Interactive delivery
The following are not applicable, and generate an error	
`-bd`	Start SMTP server
`-bm`	Deliver mail
`-bp`	Print mail queue information
`-bt`	Test configuration

In addition, any other options not specified here are not supported and are ignored with an error message.

When submitting to the local channel, *i.e.*, invoking `/pmdf/bin/sendmail`, users can set the `PMDF_FROM` environment variable to specify their From: address. PMDF will insert a `Sender:` header line with the authenticated address if it is different than the `From:` address.

17.2 Case Sensitivity of User Accounts

By default, PMDF preserves the case of addresses as originally presented to PMDF. Since UNIX accounts are case-sensitive, it is important that the mailbox of an address be in the correct case when delivering to a UNIX account. Sites with a consistent account name convention—all accounts being lowercase for instance—can want to use case insensitive matching when delivering to local mailboxes.

PMDF rewrite rules can be used to forcibly control the case of addresses going to the local channel (or any other channel), for instance, to force all local mailboxes to lowercase; see Section 2.2.5.4.

If the case of the user accounts is not regular, an alternate approach is to use PMDF aliases to achieve case insensitive matching. Note that PMDF alias matching is case-insensitive, but the case of the expanded alias is preserved. Thus for instance the use of an alias

```
adam:     AdAm
```

would mean that messages to any of "adam", "Adam", "ADAM", "aDAm", *etc.*, would be delivered to the account "AdAm".

17.3 Local Delivery on UNIX Systems

Message files queued to the `l` channel are delivered to local users by the local channel program `l_master`.

When `l_master` runs, it checks to see whether the recipient of the message has a `.forward` file in their home directory and, if so, re-enqueues the message to all recipient entries specified in that file.

For a user who does not have `.forward` file or who has a `.forward` specifying "normal" delivery, PMDF will next check if there is an entry applying to that user in the PMDF user profile database and if there is, deliver accordingly. The user profile database delivery specification, if there is one, is executed just as if it were an entry in the user's `.forward` file.

For users who need "privileged" forwarding, see also `pipe` channels, as documented in Section 26.5.

If no forwarding has been established for the user via one of the above mechanisms, then the `l` channel simply delivers to the user's directory under `/var/mail`. (In particular, note that a default entry in the PMDF profile database can be a convenient way to specify that by default deliveries should be made to an alternate location.)

17.3.1 The .forward File

The format for the `.forward` file is one or more lines, each line containing one or more comma-separated recipient entries. A recipient entry can take the following forms. When executing certain actions of a user's `.forward` actions, the `l` channel becomes that user, referred to below as *current-user*.

- *user*`@`*domain* requeues a copy of the message to the specified address. In the special case of *current-user*`@`*official-local-host-name* where *current-user* is the user on whose behalf the `.forward` file is being processed and where *official-local-host-name* is the official host name defined on the `l` channel definition, the

message is just delivered "normally" (in addition to any other forwarding specified by other entries).

- `\`*`current-user`* or *`current-user`* delivers a copy "normally" (in addition to any other forwarding specified by other entries), where *`current-user`* must be the user on whose behalf the `.forward` file is being processed.
- */directory/path/filename* appends a copy of the message to the specified file (which must be writeable by *`current-user`*, the user on whose behalf the `.forward` file is being processed).
- `+`*`/directory/path/filename`* appends a copy of the message to the specified file (which must be writeable by *`current-user`*, the user on whose behalf the `.forward` is being processed) using digest-like boundary markers between the messages.
- `|`*`command`* where *`command`* is a shell command will execute that command as *`current-user`*, the user whose `.forward` file is being processed, with standard input coming from the message being delivered.
- `|`*`command args`* where *`command args`* is a shell command with arguments will execute that command as *`current-user`*, the user on whose behalf the `.forward` is being processed, with standard input coming from the message being delivered.
- A line beginning with a `#` or `!` character is considered to be a comment line.

Example 17–1 shows a sample `.forward` file for a user jdoe who wants one copy of his messages delivered normally, one copy sent to another of his accounts, one copy sent to his pager, one copy filed in the file `thismonthsmailarchive`, and also wants to pipe the message through the `procmail` utility.

Example 17–1 Sample `.forward` File for User `jdoe`

```
\jdoe, John.Doe@system2.example.com, John.Doe@pager.example.com
/usr/users/jdoe/thismonthsmailarchive
"|/usr/bin/procmail jdoe"
```

The `RECIPIENT` environment variable is available for use in `.forward` files; it specifies the envelope recipient of the message.

17.3.2 The PMDF User Profile Database

The PMDF user profile database mechanism provides a way for PMDF system managers to provide default `.forward` file functionality for users; for instance, to specify that local users' messages should be delivered, in lieu of more specific instructions in users' own `.forward` files, to DEC MailWorks. It also provides a convenient way for users to make their own selection of any delivery format choices authorized by the PMDF system manager.

The PMDF system manager defines PMDF user profile database methods, as described below in Section 17.3.2.1, corresponding to `.forward` file style entries. Then a user can modify his or her own PMDF user profile database entry, selecting a delivery format by keyword, rather than having to correctly add an actual, possibly complicated, delivery mechanism command to his or her `.forward` file.

When executing a delivery command corresponding to a user's entry in the PMDF user profile database, PMDF becomes that user.

17.3.2.1 Configuring the PMDF User Profile Database Methods

The PMDF system manager defines delivery methods in the user profile database by first creating a PMDF profile database using a command such as

```
# su pmdf -c "pmdf crdb /dev/null PMDF_USER_PROFILE_DATABASE"
```

and then using the `pmdf profile` utility's `set method` command. The username of the user on whose behalf the command is executed will be substituted where the string `%s` is located. The substitution strings `%+`, meaning the username plus subaddress (portion after a plus character), and `%d`, meaning the user's default directory, are also available.

For example:

```
# pmdf profile
profile> set method BSD "/var/spool/mail/%s"
profile> set method DMW "|/usr/bin/inetgrecv %s"
profile> set method MIME "+/var/spool/mail/%s"
profile> show method -all
Method BSD is defined as: /var/spool/mail/%s
Method DMW is defined as: |/usr/bin/inetgrecv %s
Method MIME is defined as: +/var/spool/mail/%s
profile> exit
```

More details on the `pmdf profile` utility can be found in Section 30.2.1.

17.3.2.2 Adding User Entries to the PMDF User Profile Database

Once delivery methods have been defined, the PMDF system manager (user `root` or `pmdf`) can use the `pmdf profile` utility's `set delivery` command to set a particular user's delivery format to be one of the defined methods with the privileged `-user` qualifier, or set a default delivery format for users who do not set their own delivery format with the privileged `-default` qualifier.

For example, if the PMDF system manager has configured methods `BSD` and `DMW`, then the commands:

```
# pmdf profile set delivery BSD -user=root
# pmdf profile set delivery DMW -default
```

would direct that messages to `root` be delivered in normal UNIX mailbox format, but that all other messages delivered by the `local` channel be delivered by default using the `DMW` method.

Users can also set their own delivery format to be one of the methods defined by the PMDF system manager. For example, a user who wants to receive their messages in DEC MailWorks and whose PMDF system manager has configured `DMW` as the name of such a delivery method could use the command:

```
% pmdf profile set delivery DMW
```

17.3.3 The Option File

An option file can be used to control various characteristics of the *local* channel. This *local* channel option file must be stored in the PMDF table directory and named `l_option`, *i.e.*, `/pmdf/table/l_option`.

17.3.4 Format of the Option File

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

```
option=value
```

value can be either a string or an integer, depending on the option's requirements. If the option accepts an integer value a base can be specified using notation of the form *b*`%`*v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

17.3.5 Contents of the Option File

The available options are:

FORCE_CONTENT_LENGTH (0 or 1)

If `FORCE_CONTENT_LENGTH=1`, then PMDF adds a `Content-length:` header line to messages delivered to the *L* channel, and causes the channel not to use the `>From` syntax when `From` is at the beginning of the line. This makes local UNIX mail compatible with Sun's newer mail tools, but potentially incompatible with other UNIX mail tools.

FORWARD_FORMAT (string)

This option specifies where to find users' `.forward` files. The string `%u` means to substitute in each user's username; the string `%h` means to substitute in each user's home directory. The default behavior, if this option is not explicitly specified, corresponds to:

```
FORWARD_FORMAT=%h/.forward
```

REPEAT_COUNT (integer)
SLEEP_TIME (integer)

In case the user's new mail file is locked by another process when PMDF tries to deliver the new mail, these options provide a way to control the number and frequency of retries the local channel program should attempt. If the file can not be opened after the number

of retries specified, the messages will remain in the local queue and the next run of the `local` channel will attempt to deliver the new messages again.

The `REPEAT_COUNT` option controls how many times the channel programs will attempt to open the mail file before giving up. `REPEAT_COUNT` defaults to 30.

The `SLEEP_TIME` option controls how long in seconds the channel program waits between attempts. `SLEEP_TIME` defaults to 2 (two seconds between retries).

SHELL_TIMEOUT (integer)

The `SHELL_TIMEOUT` option can be used to control how long in seconds the channel will wait for a user's shell command in a `.forward` file to complete. Upon such time outs, the message will be returned back to the original sender with an error message along the lines of "Timeout waiting for ...'s shell command ... to complete". The default value is 600 (corresponding to 10 minutes).

SHELL_TMPDIR (directory-specification)

The `SHELL_TMPDIR` option can be used to control where the local channel creates its temporary files when delivering to a shell command. By default, such temporary files are created in users' home directories. Via this option the PMDF manager can instead choose to have the temporary files created in some other directory; *e.g.*:

```
SHELL_TMPDIR=/tmp
```

18 The Local, DECnet MAIL, and General MAIL_ Channels (OpenVMS)

The `local` channel is used to deliver messages to addresses on the local host. On OpenVMS, the `local` channel is the `l` (lowercase `L`) channel. On OpenVMS systems, this channel interfaces to the standard VMS MAIL utility. The `local` channel is also used on OpenVMS to interface PMDF to PSIMail. The conversion of PSIMail addresses to RFC 822 addresses is described in the following sections.

On OpenVMS systems, DECnet `MAIL-11` channels, `d` and `d_`, are used to deliver messages via VMS MAIL to remote hosts using the `MAIL-11` protocol. These channels differ from the local channel only in that, for messages queued to the channels, the host name in envelope `To:` addresses is significant. The `local` channel program is also used to service DECnet `MAIL-11` channels.

`MAIL_` channels are used on OpenVMS systems to provide specialized interfaces to other foreign mail transports that attach to VMS MAIL.

Many of the details of the operation of the `local` channel are described in Chapter 19 and in the *PMDF User's Guide, OpenVMS Edition*. In particular, the *PMDF User's Guide, OpenVMS Edition* describes those aspects of PMDF that are visible through the VMS MAIL user interface.

The automatic configuration generator always generates a `local` channel and, if needed, a `d` channel.

18.1 Handling VMS DECnet MAIL and PSIMail Addresses

Messages delivered to PMDF by VMS MAIL can come from the local system or can have originated on a remote system connected via either DECnet MAIL-11 or PSIMail. In the latter case, the message's return address will probably not conform to RFC 822 addressing conventions in which case PMDF must translate the address to legal RFC 822 address. PMDF uses a mapping scheme to convert VMS MAIL addresses into acceptable RFC 822 addresses and *vice versa.*

This mapping scheme is necessarily rather complex. There are four individual conversions that must be performed. These are described in detail in the following subsections.

18.1.1 Conversion of VMS To: addresses to PMDF Format

The first step in converting a VMS To: style addess to PMDF's format is to remove the foreign protocol "wrapper" from addresses of the form `IN%"address"` and then apply the translations of Table 18–1. The foreign protocol wrapper is usually, but not always, `IN%" "`.

Table 18–1 Foreign Protocol Address Mapping

Sequence	Is translated to	
’	"	(double quote)
\’	’	(single quote)
\d	"	(double quote)
\s	’	(single quote)
\D	"	(double quote)
\S	’	(single quote)
\\	\	(backslash)

These translations are necessary in order to allow users to specify double quotes inside an address even though they are not normally allowed inside a VMS MAIL address. Preferred usage when entering addresses manually is to substitute single quotes for double quotes and \’ for single quotes. For example:

```
IN%"’tony li’@hmc.edu"   maps to  "tony li"@hmc.edu
IN%"\’tonyli\’@hmc.edu"  maps to  ’tonyli’@hmc.edu
```

The next step is to apply any protocol-address-specific mapping that has been supplied. PMDF checks the mapping file to see if a mapping table named `PROTOCOL-TO-PMDF` exists. If this table exists, it is applied separately to each address inside of the foreign protocol wrapper. Specifically, a probe string of the form (note the use of the vertical bar character, `|`)

channelname|*protocolname*|*address*

is built. Here *channelname* is the name of the PMDF channel associated with the incoming mail. This will be `d` if network mail is being received and a `d` channel exists; otherwise it will be `l`. *protocolname* is the name of the foreign protocol used including the percent sign; this is usually, but not always, `IN%`. And *address* is simply the address being converted.

If a mapping entry matches the probe string, the result of the application of the mapping replaces the original addresses. If no entry matches the address is not changed in any way.

The availability of the actual protocol name used makes it possible for PMDF to handle multiple protocol names and associate different syntax rules with each one. (Of course, this is only possible if the necessary syntax modifications can be expressed in the mapping table.) For example, suppose you want to define a new foreign protocol `REVERSE%` that accepts addresses of the form `user@domain`, but the elements in domain are reversed. That is, instead of writing `IN%"user@ymir.example.com"` you would write `IN%"user@com.example.ymir"`. The following mapping would accomplish this for all domain specifications of six parts or less:

```
PROTOCOL-TO-PMDF

  *|REVERSE$%|%*@%*.*.*.*.*.*  $1$2@$9.$8.$7.$6.$5.$3$4
  *|REVERSE$%|%*@%*.*.*.*.*    $1$2@$8.$7.$6.$5.$3$4
  *|REVERSE$%|%*@%*.*.*.*      $1$2@$7.$6.$5.$3$4
  *|REVERSE$%|%*@%*.*.*        $1$2@$6.$5.$3$4
  *|REVERSE$%|%*@%*.*          $1$2@$5.$3$4
  *|REVERSE$%|%*@%*            $1$2@$3$4
```

This more complex iterative mapping will reverse domain specifications containing eight or fewer parts:

```
PROTOCOL-TO-PMDF

  *|REVERSE$%|%*@%*.*          $R{REVERSE}$1$2|$5.|$3$4
  *|REVERSE$%|%*@%*            $1$2@$3$4
  {REVERSE}*|*|%*.%*           $R{REVERSE}$0|$1$4$5.|$2$3
  {REVERSE}*|*|*               $0@$1$2
```

It would also be necessary to define a logical name for the new `REVERSE%` protocol:

```
$ DEFINE/SYSTEM/EXEC MAIL$PROTOCOL_REVERSE PMDF_SHARE_LIBRARY
```

The final step in converting a VMS To: address to PMDF's format is to convert DECnet-style addresses into RFC 822 addresses. Specifically, addresses of the form `IN%`*`system`*`::`*`user`* are converted into RFC 822 addresses of the form *`USER@SYSTEM`*. Multiple routing systems can be specified; the result will be a "percent-style" address, *e.g.,* `IN%`*`sys1`*`::`*`sys2`*`::`*`user`* will be converted to *`USER%SYS2@SYS1`*. The forced conversion of such addresses to upper case is performed by VMS MAIL and is unavoidable. Double quotes can be used to eliminate the conversion to upper case; they also make it possible to specify more than one address inside a single foreign protocol wrapper.

18.1.2 Conversion of VMS From: Addresses to PMDF Format

If the From: address is an address of the form `IN%"address"`, the `IN%" "` wrapper is removed and the same translations described in the previous section, Section 18.1.1, are performed. PMDF handles this case by effectively reversing any transformations applied to the address to make it palatable to VMS MAIL. The result should be the original address PMDF started out with. Note that this special case does arise in practice when incoming PMDF messages are forwarded through VMS MAIL back out to PMDF.

If the address is not something PMDF recognizes as one of its own, the first step in the rewriting process is to attempt to use any special mapping that has been supplied. PMDF checks the mapping file to see if a mapping table named VMSMAIL-TO-PMDF exists. If this table exists it is applied to each address. Specifically, a probe string of the form

`channelname``|`*`address`*

is constructed. Here *`channelname`* is the name of the PMDF channel associated with the incoming mail. This will be `d` if network mail is being received and a `d` channel exists; otherwise it will be "l". *`address`* is simply the address being converted.

If no entry in the mapping table matches the probe string, the conversion process continues as if no mapping table was given; the address is not changed. If an entry does match, the result of the application of the mapping replaces the original addresses. This resultant address can either be a VMS MAIL address or a RFC 822 address. The mapping template should specify a $< metacharacter if it produces a VMS MAIL address and $> metacharacter if it produces a RFC 822 address. No further conversion will be done if $> is specified.

The following example shows a very simple mapping that converts PSIMail addresses into (possibly legal) domain addresses:

```
VMSMAIL-TO-PMDF

  *|PSI$%1::*          $1@one.psi.company.com$>
  *|PSI$%2::*          $1@two.psi.company.com$>
  *|PSI$%*::*          $2@$1.psi.company.com$>
```

Now, if the conversion process has resulted in a VMS MAIL address at this point (which includes addresses generated by DECnet MAIL, PSIMail, or various other mail systems), it is converted into RFC 822 quoted string format and the local host name is appended to the address. DECnet addresses are handled in a special manner — double colons are converted into percent signs and the halves of the address are swapped. The special case of a foreign protocol address that does not require RFC 822 quoting is handled by doubling the percent signs in the address (effectively quoting them). For example:

```
IN%"dan@sc.cs.cmu.edu"  maps to  dan@sc.cs.cmu.edu
IN%"'test 1'@tester"    maps to  "test 1"@tester
NED                     maps to  ned@local-host
A::B::C::D              maps to  d%c%b%a@local-host
PSI%1002::TEST          maps to  "PSI%1002::TEST"@local-host
DSIN%48374334343        maps to  DSIN%%48374334343@local-host
```

18.1.3 Conversion of PMDF From:, To:, and Cc: Addresses to VMS Format

First, the `useresent` channel keyword, if used on the `local` channel, controls whether or not PMDF preferentially uses any Resent- headers in its construction of headers to pass to VMS MAIL. Next, PMDF checks for the existence of a `PMDF-TO-VMSMAIL` table in the mapping file. If this table exists, each address is converted into a probe string of the form

channelname|*address*

where *channelname* is the name of the channel performing the operation and *address* is the address being processed. Since this operation is normally done when PMDF is delivering messages to VMS MAIL, the channel name should be the name of the channel actually delivering the message.

If no mapping entry matches the probe string, the address is not changed. However, if an entry does match, the result of the mapping replaces the address. The result can either be a VMS MAIL address or a RFC 822 address. The mapping template should

specify a $< metacharacter if it produces a RFC 822 address and $> metacharacter if it produces a VMS MAIL address. No further conversion will be done if $> is specified.

If the conversion process has resulted in a RFC 822 address, at this point, the process proceeds to convert the address into a format acceptable to VMS MAIL. This operation is almost the inverse of conversion described in Section 18.1.1, Conversion of VMS To: addresses to PMDF Format. The "\d" and "\s" forms are substituted, respectively, for double and single quotes. An `IN%" "` wrapper is added to each address.

A couple of additional operations are also performed in an attempt to make up for limitations of VMS MAIL. VMS MAIL makes some attempt to add DECnet routing information to the lines it processes. Specifically, the name of the local DECnet node is prepended to some addresses. This information is added because without it addresses transferred to remote nodes via `MAIL-11` are not repliable (the operation is actually performed by the remote node using its own name for the originating node).

However, the processing done by VMS MAIL is incomplete. First, if multiple addresses appear on the VMS MAIL From: line, only the first address has DECnet routing information added to it. Second, the VMS MAIL `To:` and `cc:` lines are also used for replies in some cases. In particular, the Pathworks MAIL application provides a `REPLY-TO-ALL` function that attempts to send to all addresses on the VMS MAIL `From:`, `To:` and `cc:` lines. Unfortunately VMS MAIL fails to prepend correct node information to `To:` and `cc:` line addresses.

PMDF attempts to correct these problems if requested to do so. The `daemon` channel keyword is used to activate this feature — if this keyword is present the argument given to it is interpreted as a DECnet node name (the double colons can be included if desired; they will be added automatically if they are not specified as part of the node name). This node name information is prepended to the second and subsequent addresses appearing on the VMS MAIL From: line. This node information is appended to *every* address that appears on the `To:` and `cc:` lines.

If the special argument `SYS$NODE` is given to the `daemon` keyword the `SYS$NODE` logical is translated and the result is used as the prepended node name. This can be useful in heterogeneous cluster environments with complex queue setups where the DECnet node name associated with a channel can not be known before the delivery job starts running in a specific queue.

This sort of `daemon` keyword can be used on any channel associated with VMS MAIL: local `l`, DECnet MAIL-11 `d`, `d_`, and MAIL `mail_` channels are all affected. Usually this feature is used with d and d_ channels; it can occasionally be useful on the `l` channel, but a price is incurred in terms of unnecessarily complex addresses in this case. Note that the `daemon` keyword controls additional functionality when used in conjunction with mail_ channels; see the Section 18.1.4 for additional details.

18.1.4 Conversion of PMDF Envelope To: Addresses to VMS Format

This is by far the most complex case. The actual conversion performed is channel-specific.

- If the PMDF component doing the conversion is associated with the local, l, channel the local host part of the address "`@`*`local-host`*" is removed and then the inverse of the conversion described in Section 18.1.2 is performed. There are three additional twists, however.

  First of all, addresses of the form *`A%B`* are converted to the form *`B::A`* and addresses of the form *`A!B`* are converted to *`A::B`*. Double `%s` are converted to a single percent and the ordering is left unchanged. This is done so that literal percent signs can be specified in an unquoted address. However, if the mailbox is quoted, the string is "dequoted" according to the rules of RFC 822 and the result is used without further translation. These extra twists provide support for complex DECnet MAIL routing. For example:

```
NED@local-host                  maps to  NED
NED%YELLOW@local-host           maps to  YELLOW::NED
NED%YELLOW.RED@local-host       maps to  YELLOW.RED::NED
"YELLOW::NED"@local-host        maps to  YELLOW::NED
"A::B::C::D"@local-host         maps to  A::B::C::D
D%C%B%A@local-host              maps to  A::B::C::D
G%F.E%D.C%B.A@local-host        maps to  B.A::D.C::F.E::G
"PSI%1002::TEST"@local-host     maps to  PSI%1002::TEST
A!B@local-host                  maps to  A::B
B!C%A                           maps to  A::B::C
C%A!B                           maps to  A!B::C
DSIN%%3784374343434             maps to  DSIN%3784374343434
```

- If the address is associated with the DECnet MAIL-11 channel `d` or `d_` the same operations are performed except that no "`@`*`local-host`*" part is removed and at signs are handled just like percent signs. For example:

```
NED@node                  maps to  NODE::NED
NED%YELLOW@node           maps to  NODE::YELLOW::NED
NED%YELLOW.RED@node       maps to  NODE::YELLOW.RED::NED
D%C%B%A@node              maps to  NODE::A::B::C::D
G%F.E%D.C%B.A@node        maps to  NODE::B.A::D.C::F.E::G
"PSI%1002::TEST"@node     maps to  NODE::PSI%1002::TEST
DSIN%%3743743@node        maps to  NODE::DSIN%3743743
```

- If the address is associated with a `mail_`*`xxx`* channel, the address is converted in the same way as it would be for a DECnet MAIL-11 channel, but in addition the address is prefixed with the string "*`xxx`*`%`". This mechanism, along with channel table rewriting, can be used to make the rewrite rules for PSImail addresses much simpler.

- A completely different strategy is used if the daemon keyword is specified in the definition of a mail_ channel. If daemon is specified, the official host name associated with the channel is used as a prefix and the address is enclosed in double quotes and used as a suffix. Note that the argument to the daemon keyword is significant; see Section 18.1.2 for details.

  For example, suppose the daemon keyword is applied to the mail_psi channel whose official hostname is PSI%xyz::IN%. The address *user*@x.y would then be converted to PSI%xyz::IN%"*user*@x.y". This mechanism makes it possible for PMDF to send messages to another PMDF system via any mechanism supplied by VMS MAIL.

- There's one final feature available — if the channel is marked with the 733 keyword all system names are truncated at the first period that appears in their names. This makes it possible to eliminate lots of channel specific rewrite rules in some configurations. For example, if the d channel was marked as a 733 channel, the addresses shown in the second bulleted item above would be converted as follows:

```
NED@node              maps to  NODE::NED
NED%YELLOW@node       maps to  NODE::YELLOW::NED
NED%YELLOW.RED@node   maps to  NODE::YELLOW::NED
D%C%B%A@node          maps to  NODE::A::B::C::D
G%F.E%D.C%B.A@node    maps to  NODE::B::D::F::G
```

18.2 Accessing Remote OpenVMS DECnet MAIL and PSIMail Systems

To provide access to remote DECnet MAIL and PSIMail via the local channel, versions of PMDF prior to version 3.2 used rewrite rules of the form

```
red.okstate.edu       $U$%RED@yellow.okstate.edu
green.okstate.edu     PSI$%101202::$U@yellow.okstate.edu
blue.okstate.edu      $U$%BLUE@yellow.okstate.edu
slate.okstate.edu     $U$%GRAY@yellow.okstate.edu
```

or of the form

```
red.okstate.edu       RED::$U@yellow.okstate.edu
green.okstate.edu     PSI$%101202::$U@yellow.okstate.edu
blue.okstate.edu      BLUE::$U@yellow.okstate.edu
slate.okstate.edu     GRAY::$U@yellow.okstate.edu
```

Here RED and BLUE are remote DECnet systems, GREEN is a remote PSIMail system, GRAY is a remote DECnet system whose domain name (slate.okstate.edu) differs from its DECnet node name, and YELLOW is the local host. This mechanism is still available in order to preserve compatability with earlier versions.

The problem with this scheme is that while these rewrite rules provide full access to the remote systems, they also rewrite addresses in message headers, resulting in undesirable header address formats.

Enhanced rewriting facilities introduced with PMDF V3.2 make it possible to avoid this problem and still provide the necessary access to remote systems. This is accomplished as follows:

1. Use the rewriting rules **only** to canonicalize addresses, that is, the rewrite rules convert all possible incoming addresses into the proper domain format addresses that should appear in the message headers.
2. Use the DECnet MAIL-11 channel (`d` channel) to provide access to remote DECnet systems. Use special MAIL channels to provide access to PSIMail.
3. Use appropriate channels in conjunction with channel-level translation operations (in the channel block) to map domain address back into DECnet MAIL or PSIMail addresses. If possible mark the channel with the `733` keyword and use domain names whose leftmost section (the part before the first period) corresponds to the DECnet MAIL or PSIMail system name in order to lessen the number of channel-level translation operations that are needed.

For example, the rewrite rules shown above could be implemented in PMDF with the following rewrite rules and channel blocks:

```
red.okstate.edu       $U@RED.OKSTATE.EDU
red                   $U@RED.OKSTATE.EDU
green.okstate.edu     $U@GREEN.OKSTATE.EDU
green                 $U@GREEN.OKSTATE.EDU
blue.okstate.edu      $U@BLUE.OKSTATE.EDU
blue                  $U@BLUE.OKSTATE.EDU
slate.okstate.edu     $U@SLATE.OKSTATE.EDU
slate                 $U@SLATE.OKSTATE.EDU
gray                  $U@SLATE.OKSTATE.EDU

...

d
DECNET-MAIL
RED.OKSTATE.EDU       RED
BLUE.OKSTATE.EDU      BLUE
SLATE.OKSTATE.EDU     GRAY

mail_psi
PSI-MAIL
GREEN.OKSTATE.EDU     101202
```

Or, using the `733` keyword to simplify things somewhat:

```
red.okstate.edu       $U%RED.OKSTATE.EDU@DECNET-MAIL
red                   $U%RED.OKSTATE.EDU@DECNET-MAIL
green.okstate.edu     $U@GREEN.OKSTATE.EDU
green                 $U@GREEN.OKSTATE.EDU
blue.okstate.edu      $U%BLUE.OKSTATE.EDU@DECNET-MAIL
blue                  $U%BLUE.OKSTATE.EDU@DECNET-MAIL
slate.okstate.edu     $U@SLATE.OKSTATE.EDU
slate                 $U@SLATE.OKSTATE.EDU
gray                  $U@SLATE.OKSTATE.EDU

...

d 733
DECNET-MAIL
SLATE.OKSTATE.EDU      GRAY
```

```
mail_psi
PSI-MAIL
GREEN.OKSTATE.EDU    101202
```

18.3 File Attachments and Pathworks MAIL for PCs

Pathworks MAIL for PCs can generate file attachments. However, the format used is completely nonstandard and specific to Pathworks MAIL for PCs. Only one attachment is supported by Pathworks.

18.3.1 Pathworks Mail to MIME

Note: The following discussion applies only to OpenVMS systems. This functionality is not supported on UNIX platforms.

PMDF provides facilities to convert such attachments into MIME body parts. However, these facilities need some sort of flag so they know what messages need to be processed. Unfortunately no convenient flag (*e.g.,* a distinguishing header line) is provided by Pathworks MAIL itself, so a flag had to be chosen that could be associated with incoming Pathworks MAIL messages.

The flag PMDF has chosen is a special "X-Mailer:" header line. The contents of this header line must be either "Pathworks MAIL V4.0" or "Pathworks MAIL V4.1". Header lines of this sort tell PMDF that it is faced with a message that can contain one or more file attachments. PMDF can then be told to scan this message and convert it into MIME format.

The most convenient way to add this header line to messages coming from Pathworks MAIL for PCs is to define the `PMDF_HEADER` logical appropriately. This can be done automatically in the context of the Pathworks MAIL server process by adding the following definition either to the appropriate user `LOGIN.COM` files or to the system-wide `SYLOGIN.COM` file (if everyone uses the same version of Pathworks MAIL):

```
$ if f$mode() .eqs. "NETWORK" then -
    if f$locate("PCSA$MAIL",f$logical("SYS$NET")) .lt. -
       f$length(f$logical("SYS$NET")) then -
         define pmdf_header "X-Mailer: Pathworks Mail V4.1"
```

This definition is conditionalized so it only applies to Pathworks MAIL servers. Any messages sent to PMDF will then have the appropriate "X-Mailer:" header.

It is also necessary to instruct PMDF to convert these messages to MIME format. This is done by adding a special channel to your configuration to associate with incoming Pathworks MAIL and then adding instructions to request MIME conversions for this channel to your mapping file.

The special channel block for incoming Pathworks MAIL should look like this:

```
d_pathworks
PATHWORKS-MAIL
```

The channel block is all that's needed; this channel is only used as a placeholder and needs no rewrite rules.

The next step is to add appropriate instructions to the CHARSET-CONVERSION table in your mapping file to activate MIME format conversion. A single entry of the form

```
CHARSET-CONVERSION

  IN-CHAN=d_pathworks;OUT-CHAN=*;CONVERT                Yes
```

will activate the conversion procedure for all messages entering PMDF via the Pathworks MAIL channel. The CHARSET-CONVERSION mapping is described in Chapter 6.

18.3.2 MIME to Pathworks Mail

When sending mail to Pathworks Mail users, it is possible to have PMDF write the message using Pathworks' own attachment format. Remember only one attachment is supported by Pathworks Mail, other attachments in the same message will not be visible when viewing Pathworks Mail.

The only step is to add appropriate instructions to the CHARSET-CONVERSION table in your mapping file to activate conversion. A single entry of the form

```
CHARSET-CONVERSION

  IN-CHAN=*;OUT-CHAN=l;CONVERT                  Pathworks
```

will activate the conversion procedure for all messages entering the local channel. The Pathworks attachment format is used under the MIME structure, so the message is still in proper MIME format for the local users not using Pathworks Mail.

19 The PMDF User Interface on OpenVMS

Much of the information provided in this chapter is intended to supplement that found in the first two chapters of the OpenVMS edition of the *PMDF User's Guide*.

User interaction with PMDF takes place via any of several interfaces including: PMDF MAIL, PMDF Pine, VMS MAIL, DECwindows MAIL, Pathworks MAIL, GoldMail, and MINT. If PMDF-MR is available, then users can also utilize ALL-IN-1 IOS, MailWorks clients (A1MAIL) and other MAILbus (Message Router) agents available from Hewlett-Packard Company. In addition, programs which utilize the callable interface to VMS MAIL should work with PMDF. Users do not directly interact with PMDF and, for the most part, should be unaware of PMDF's existence.

If the optional PMDF-MR product has been installed, ALL-IN-1 IOS and MailWorks users can direct messages through PMDF by specifying the name of the PMDF Message Router mailbox, usually "PMDF", as a route in the address.

The standard capabilities of ALL-IN-1 IOS and MailWorks are supported by PMDF-MR, including multiple attachments, non-textual bodyparts, and delivery and read receipt notification. In addition, PMDF-MR can be configured to perform automatic conversion of WPS-Plus and DX format messages to standard ASCII text.

19.1 Sending Mail with VMS MAIL

This section provides additional information which supplements that found in the section "Sending mail with VMS MAIL" in the first chapter of the *PMDF User's Guide*.

In VMS MAIL, whenever a message is sent with the IN% protocol prefix in its address, VMS MAIL activates PMDF's MAIL-11 foreign protocol interface to hand over the message to PMDF. PMDF receives the message and places the message in the proper PMDF channel queue for disposition.

19.1.1 Using a Prefix Other than IN%

PMDF supports the use of protocol name prefixes other than IN%. In fact, PMDF supports the use of more than one prefix at a time.

The prefix or prefixes that activate PMDF are controlled by logical names of the form MAIL$PROTOCOL_*x*, where *x* is the prefix name. All such logical names should translate to the same string to which MAIL$PROTOCOL_IN translates:

```
$ DEFINE/SYSTEM/EXEC MAIL$PROTOCOL_FOO 'F$TRNLNM("MAIL$PROTOCOL_IN")'
```

The installation procedure for PMDF only defines the system-wide logical name MAIL$PROTOCOL_IN. Additional logical names can be defined as needed but this definition should *not* be removed; it should be possible to use the standard IN% protocol name at every site running PMDF.

PMDF must choose a specific protocol name prefix to place in From: addresses. This prefix is controlled by the logical name PMDF_PROTOCOL. Since actual delivery of messages to VMS MAIL is done using system batch jobs, the only sensible place to define this logical name is in the system logical name table.

For example, the following definition would tell PMDF to use the BITNET% prefix in its From: addresses instead of IN%:

```
$ DEFINE/SYSTEM PMDF_PROTOCOL "BITNET%"
```

When defined, the translation value of the PMDF_PROTOCOL logical must be terminated by a percent sign, %. The PMDF_PROTOCOL logical name is not defined by default. In the absence of a definition PMDF uses the standard IN% protocol name.

Note: If PMDF_PROTOCOL is defined to be "FOO%" a corresponding MAIL$PROTOCOL_FOO logical *must* be defined or messages will not be repliable. In general it is not a good idea to use a protocol name other than IN%. The use of IN% represents a standard of sorts; lots of sites support it. You can think you have a better choice, but remember that choosing a different name can lead to considerable confusion in the long run.

19.1.2 Displaying a Welcome Message When PMDF is used

Under some circumstances it can be useful to have PMDF display a welcoming message when a user invokes PMDF via PMDF MAIL or VMS MAIL. Such messages can contain information about the status of the mail system, such as "Our network is down until tomorrow", "The name of system A has been changed to B", "System A is being upgraded and won't be reachable until next week", and so forth.

When PMDF is activated by PMDF or VMS MAIL through the use of a PMDF address (*i.e.,* an address that activates the PMDFSHR image), it attempts to translate the logical name PMDF_WELCOME. If this logical name is not defined (which is normally the case) PMDF will not display anything. If this logical name does translate to some equivalence string, PMDF first checks to see if the first character is an underscore. A leading underscore instructs PMDF to display a welcome message before each and every message is sent. If the underscore is not present the welcome message is only displayed initially; subsequent displays are suppressed unless the value of the logical name changes in some way. The leading underscore, if present, is removed.

PMDF then checks to see if the next character in the string is an at sign. If it is, the remainder of the string is treated as a file name; PMDF opens the file and copies it to the user's terminal. No special privileges are used to open the file; PMDF will not display anything if the file cannot be opened. This makes it possible to use ACLs to protect the file so that only a subset of the user community will see the message. If the next character of the string is not an at sign, PMDF simply displays the string on the user's terminal.

No special privileges beyond those the user normally has are used for any of these operations. The logical name can be defined in the process, job, group or system logical name tables and the mode of the logical name is not relevant.

19.1.3 Sending Binary Files with SEND/FOREIGN

PMDF supports the SEND/FOREIGN facility of VMS MAIL. A command of the form

```
MAIL> SEND/FOREIGN file-spec
```

will send the file *file-spec* in foreign format. The file parameter *file-spec* is required. Note that this command is not documented in the VMS MAIL documentation.

PMDF detects the /FOREIGN format request and acts accordingly — the file is read using block mode I/O and encoded in a printable form for transmission. The resulting message preserves all file attributes and can be used to send any kind of OpenVMS file, including indexed files, without losing any information.

Upon reception of such messages, PMDF will detect the use of foreign encoding and will undo the encoding used prior to delivering the message to VMS MAIL in foreign format. The resulting message cannot be read in VMS MAIL (unless the file sent was a DDIF, DTIF, or DOTS file); it must be extracted with the VMS MAIL EXTRACT command to a file before it can be used. A DDIF, DTIF, or DOTS file will be displayed to the maximum extent possible — DECwindows MAIL will use the CDA viewer to display the document while regular VMS MAIL will only display the text components of the file content.

There is no place to store additional header information for such messages, so PMDF delivers the headers for these messages in a separate message after the foreign format message. This delivery mechanism can be disabled with the `noforeign` channel keyword; the `foreign` channel keyword is the default.

19.1.4 Header Lines in Messages

PMDF uses two sources of information to construct message headers for messages it receives from VMS MAIL. The first is VMS MAIL itself, which provides a To:, From:, Cc:, and Subject: line, and a list of addressees. The second source of information consists of logical names which you can define yourself. The following logical names can be defined to create header lines:

- PMDF_COMMENTS (*)
- PMDF_DELIVERY_RECEIPT_TO
- PMDF_ERRORS_TO (*)
- PMDF_FROM
- PMDF_HEADER
- PMDF_READ_RECEIPT_TO
- PMDF_REFERENCES (*)
- PMDF_REPLY_TO (*)
- PMDF_RESENT_FROM
- PMDF_RESENT_REPLY_TO

- PMDF_HEADER_n
- PMDF_IMPORTANCE (*)
- PMDF_KEYWORDS (*)
- PMDF_ORGANIZATION (*)
- PMDF_PRIORITY (*)
- PMDF_SENSITIVITY (*)
- PMDF_TIMEZONE
- PMDF_WARNINGS_TO (*)
- PMDF_X_FAX_DEFAULTS (*)
- PMDF_X_PS_QUALIFIERS (*)

The logical names marked with an asterisk, *, are described in the *PMDF User's Guide*; the remaining logicals and associated header lines are described below.

The PMDF_TIMEZONE logical name is set by the system manager in the system logical name table to translate to the system's local timezone; users can only override this setting if no system PMDF_TIMEZONE logical is set. None of the other logical names are set by default, and even if they are, users can define their own overriding values to customize the headers attached to the messages they send.

When changing the value of the PMDF_TIMEZONE logical, note that this logical is normally defined at system startup by the `SYS$STARTUP:pmdf_startup.com` procedure created when PMDF was installed; see the OpenVMS edition of the *PMDF Installation Guide*.

After changing the value of the PMDF_TIMEZONE logical, the Dispatcher should be restarted so that services under the Dispatcher will be made aware of the change and properly use the new value.

Note: If the PMDF_TIMEZONE logical does not exist (for example if you choose to deassign it in `pmdf_com:pmdf_site_startup.com`), then PMDF looks for the following system logicals to determine the local time zone. First it looks for SYS$LOCALTIME, then if that is not defined, it looks for SYS$TIMEZONE_DIFFERENTIAL and SYS$TIMEZONE_NAME.

19.1.4.1 Cc: Header Lines

Cc: header lines are produced in outgoing mail when a recipient address is passed to PMDF by VMS MAIL and the address can be matched with the text that appears on the VMS MAIL Cc: line. The addresses on the VMS MAIL Cc: line itself cannot be used directly; they are not suitable for use on a RFC 822 Cc: line and are used to construct an X-VMS-Cc: line instead.

VMS MAIL provides no indication of where the envelope addresses it passes to PMDF came from (To: line, Cc: line, or both). PMDF tries to match the address with the contents of the X-VMS-To: and X-VMS-Cc: lines, but this attempt is necessarily imperfect and can fail. If it does fail and PMDF cannot localize an address it will be placed on the To: line by default. The result is that addresses that originated on the VMS MAIL Cc: line can end up on the outgoing message's To: header.

19.1.4.2 Content-transfer-encoding: Header Lines

PMDF will insert a Content-transfer-encoding: header that describes the encoding applied to the data in outgoing messages. This is normally 7BIT for messages containing only 7 bit text data and 8BIT for messages containing 8 bit text data. Messages sent with SEND/FOREIGN will normally be encoded using the MIME-compliant BASE64 encoding and will be labelled accordingly. The Content-transfer-encoding: header is defined by MIME (RFCs 2045-2049); it is not part of RFC 822.

19.1.4.3 Content-type: Header Lines

PMDF will insert a Content-type: header into outgoing messages that describes the type of data being sent. Most messages sent from VMS MAIL will be labelled as "text/plain" along with whatever character set is specified by the `charset7` or `charset8` channel keywords on the local channel. Files sent with SEND/FOREIGN will be labelled as "application/vms-rms" if no special RMS semantics are attached. Currently the only RMS semantics that are recognized are DDIF, DOTS, and DTIF; these are labelled as "application/ddif", "application/dots", and "application/dtif" respectively. In any of these cases the Content-type: header will also contain a VMS-FDL parameter containing an FDL description of the file.

When a message labelled as "application/vms-rms" or any of the special semantic tags is received by VMS MAIL the encoding will be reversed and the stored FDL information will be applied to the message as it is delivered. The result will be a foreign format message whose file attributes and contents are preserved.

19.1.4.4 Delivery-receipt-to: Header Lines

The use and construction of Delivery-receipt-to: headers is discussed in Section 19.3.

19.1.4.5 Disposition-notification-to: Header Lines

The use and construction of Disposition-notification-to: headers is discussed in Section 19.3.

19.1.4.6 From: and Sender: Header Lines

The From: header is constructed by translating the logical name PMDF_FROM. If it is defined, its equivalence string is used as the From: address for the message and the authenticated address of the person sending the message is placed on a Sender: line in the message header.

If the PMDF_FROM line does not translate to anything (by default it will not), the address of the person sending the message is used as the From: address and no Sender: line is inserted in the message header.

The setting of PMDF_FROM is ignored when messages pass through VMS MAIL because of automatic forwarding. Addresses specified by the PMDF_FROM logical undergo normal PMDF address rewriting; they are not exempt from rewrite processing.

If PMDF_FROM translates to a string beginning with and ending in a question mark "?", PMDF will take the enclosed string and prompt the user with it. Whatever the user types will be placed on the From: header line. This prompt will appear after any message has been entered.

Prompting is only possible in regular command-oriented VMS MAIL; it cannot be done in DECwindows MAIL. PMDF checks to make sure that prompting is possible and will ignore any requests for prompts in environments that don't support it.

19.1.4.7 Read-receipt-to: Header Lines

The use and construction of Read-receipt-to: headers is discussed in Section 19.3.

19.1.4.8 Resent-date: Header Lines

A Resent-date: line is created when a message being delivered by PMDF to VMS MAIL is forwarded back to PMDF by some type of VMS MAIL forwarding (usually a SET FORWARD to an IN% address). Such a message already has a Date: header, so PMDF adds a Resent-date: instead. The format of a Resent-date: header is the same as a Date: header. If both a Date: and Resent-date: header are already present no time stamp information is added apart from the Received: line (multiple Resent-date: headers are not legal RFC 822 syntax).

19.1.4.9 Resent-from: Header Lines

Resent-from: headers and the associated PMDF_RESENT_FROM logical are handled in the same way as From: and PMDF_FROM, with the exception that forwarding does not affect these headers in the same way that it affects From: headers.

19.1.4.10 Resent-reply-to: Header Lines

Resent-reply-to: headers and the associated PMDF_RESENT_REPLY_TO logical are handled in the same way as Reply-to: and PMDF_REPLY_TO, with the exception that forwarding does not affect these headers in the same way that it affects Reply-to: headers.

19.1.4.11 Resent-to: Header Lines

A Resent-to: line is created when a message being delivered by PMDF to VMS MAIL is forwarded back to PMDF by some type of VMS MAIL forwarding (usually a SET FORWARD to an IN% address) and the message already has a To: header. In this case the text that would normally be placed on the To: header is placed on the Resent-to: header line instead.

No logical name is provided to set this header.

19.1.4.12 Subject: Header Lines

The Subject: header line is obtained directly from VMS MAIL (which has either constructed it itself or gotten it from the user).

19.1.4.13 To: Header Lines

To: header lines are produced in outgoing mail when a recipient address is passed to PMDF by VMS MAIL and the address can be matched with the text that appears on the VMS MAIL To: line. The addresses on the VMS MAIL To: line itself cannot be used directly; they are not suitable for use on the To: line and are used to construct an X-VMS-To: line instead.

VMS MAIL provides no indication of where the envelope addresses it passes to PMDF came from (To: line, Cc: line, or both). PMDF tries to match the address with the contents of the X-VMS-To: and X-VMS-Cc: lines, but this can fail. If it does fail PMDF places the address on the To: line by default. The result is that addresses that originated on the VMS MAIL Cc: line can end up on the outgoing message's To: header.

19.1.4.14 X-Envelope-to: Header Lines

The X-Envelope-to: header line contains a duplicate of the address that appears in the message envelope. This header line is used by PMDF to reconstruct envelope addresses that have been damaged by various transport mechanisms (notably BITNET and NJE, which truncate envelope addresses to eight characters). The X-Envelope-to: line for a particular copy of a message contains only the address that that particular copy is being sent to; it does *not* contain all the addresses on the To: line (except in the simplest case where only a single copy of the message is needed).

The X-Envelope-to: line is not user-settable. Its presence or absence in a particular copy of a message is controlled by the `x_env_to`, `single`, and `nox_env_to` keywords on the corresponding channel; see Section 2.3.4.61.

X-Envelope-to: lines are unique among header lines in that they are completely replaced each time a message is enqueued by PMDF. (Most other header lines are cumulative.) An X-Envelope-to: header line only reflects the most recent envelope destination in the case of forwarding. The diagnostic usefulness of an X-Envelope-to: header line has been almost entirely superseded by the use of "received for" clauses

in Received: header lines and NOTARY notification messages including final-recipient information.

Note: X-Envelope-to: header lines are an extension to RFC 822.

19.1.4.15 X-VMS-Cc: Header Lines

VMS MAIL's Cc: line (which is not in RFC 822 format) is placed on the X-VMS-Cc: line of the message header. See Section 19.1.4.1.

Note: X-VMS-Cc: header lines are an extension to RFC 822.

19.1.4.16 X-VMS-To: Header Lines

VMS MAIL's To: line (which is not in RFC 822 format) is placed on the X-VMS-To: line of the message header. See Section 19.1.4.13.

Note: X-VMS-To: header lines are an extension to RFC 822.

19.1.5 Message Headers on Forwarded Messages

Messages sent via a SET FORWARD to PMDF are for the most part handled in the same way as messages sent to PMDF directly. An exceptional case arises, however, when PMDF delivers a message to a local VMS MAIL address that is in turn forwarded back to PMDF via a SET FORWARD directive. PMDF detects this case and handles it as a special type of forwarding. In particular:

1. The message RFC 822 header already attached to the message is kept as message header and modified. In contrast, manually forwarded messages receive two headers — the original header, which becomes part of the body of the message, and a new header which is added as the message is forwarded. This handling applies regardless of how PMDF has been configured to handle message headers during local delivery (that is, the `headerbottom` and `headeromit` channel keywords have no effect on forwarding).
2. Since such messages always have a Date: header already attached, a Resent-Date: header is added instead of a Date: header. No time stamp header is added if the message had both a Date: and a Resent-Date: header already. (Note that a Received: header is always added and this header does provide some time stamp information.)
3. A From: header is only attached if the message does not have one already. Resent-from: headers are never added.
4. A Resent-To: is added if a To: header is already present; a To: header is added if no To: header is present in the original message.

5. Although the code to add user-specified headers is executed, it has no effect since this type of forwarding is handled at a system level; it is not handled under the control of the user's account. System-wide settings of user-specified headers will be applied, however.

19.1.6 Temporary File Storage

Both PMDF and VMS MAIL use a number of temporary files to construct messages. These files are normally created in the SYS$SCRATCH directory. This applies to all VMS MAIL variants including PMDF MAIL, VMS MAIL, and DECwindows MAIL, but is especially important when using callable MAIL. Callable MAIL will fail under some circumstances if SYS$SCRATCH is not defined. (Specifically, callable MAIL will fail if the message is written one record at a time; this operation requires an intermediate file for storage.)

VMS MAIL creates named temporary files in whatever directory SYS$SCRATCH points to. These files generally have a `.tmp` extension and the word `mail` somewhere in the name. PMDF creates unnamed temporary files on the device SYS$SCRATCH that are not entered into any directory. Note that in either case the files are owned by the user creating them and the user must have the necessary quotas to create the file or files. No special PMDF or VMS MAIL privileges are employed to access files in SYS$SCRATCH.

PMDF can be directed to use an alternate scratch area by defining the PMDF_SCRATCH logical name. If PMDF_SCRATCH is defined PMDF will create its unnamed temporary files there. This applies to all of PMDF, not just the parts that run under VMS MAIL. If PMDF_SCRATCH is undefined or unusable PMDF will then try SYS$SCRATCH, and if SYS$SCRATCH is undefined or unusable the device associated with the current default directory will be used. If this device does not exist or is unusable PMDF will try to use SYS$LOGIN.

VMS MAIL does not use PMDF_SCRATCH for its temporary files under any circumstances, of course.

19.1.7 DECwindows MAIL and Account Quotas

DECwindows MAIL consumes large amounts of various quotas. PMDF also consumes some resources above and beyond those used by regular VMS MAIL. The result is that low account quotas can cause the combination of PMDF and DECwindows MAIL to fail. In particular, a FILLM quota of at least 200 is needed for proper operation under OpenVMS V5.3 or later. An ENQLM of at least 400 is strongly recommended as well.

19.1.8 Handling VMS MAIL Errors

VMS MAIL and PMDF must cooperate closely in their handling of errors detected during message interchange. PMDF is careful to inform VMS MAIL about any errors it detects while VMS MAIL is sending a message to PMDF. Conversely, PMDF tries to handle the various error conditions that can arise while delivering a message to VMS MAIL.

PMDF categorizes errors returned by VMS MAIL as either temporary or permanent. A temporary error is something like "node down". Such a problem might be corrected in the future. A permanent error is one like "no such user" which is unlikely to be rectified in the future. PMDF treats the following errors as permanent errors:

NOSUCHNODE — Specified DECnet node does not exist
NOSUCHUSR — No such user exists
USERDSABL — Specified user cannot receive new mail
SYNTAX — Syntax error in username/node specification
TEXT — Error occurred during MAIL-11 message delivery.

All other errors are considered to be temporary.

In the case of a temporary error PMDF simply aborts its attempt to send the current message. Periodic attempts will be made to send the message until the error condition disappears and the message makes it through. (Note that if the error condition is never remedied the message will eventually "expire" and will automatically be returned to the sender.)

In the case of a permanent error, PMDF records both the bogus address and the error it generated. PMDF then dispatches a notification of the error, along with a copy of the message, to the sender. An additional copy of the error notice is mailed to the local postmaster (usually, but not always, the system manager) by default. The sending of failed mail notices to the local postmaster can be disabled by adding the `nosendpost` keyword to the local channel block in the PMDF configuration file. Return of message contents to the postmaster can be restricted by adding the `postheadonly` keyword to the local channel.

19.1.9 Accepting MIME Headers from VMS MAIL 7.2 or Later

As of OpenVMS 7.2, VMS MAIL is capable of generating MIME headers. PMDF does not normally expect to see MIME headers coming from VMS MAIL, and will normally ignore such headers if they are present. The special logical PMDF_MAIL_MIME_HEADERS can be used to instruct PMDF to read MIME headers from the top of the message body coming from VMS MAIL. If PMDF_MAIL_MIME_HEADERS translates to `TRUE`, then PMDF will properly interpret MIME headers on messages originating from VMS MAIL.

PMDF_MAIL_MIME_HEADERS can be defined either system-wide, or by individual users.

19.2 Receiving Mail in VMS MAIL

RFC 822 specifies a large number of fields that can appear in the header of a message. These include familiar things like “From:”, “To:”, “Subject:”, and “Return-Path:” and obscure things like “Encrypted:”, “Resent-Message-ID:”, and “Resent-reply-to:”. Unfortunately, VMS MAIL only provides for “From:”, “To:”, “Cc:”, and “Subj:” information in its message headers. This leaves PMDF with two choices: either discard header lines that do not correspond to VMS MAIL header lines, or insert them somewhere in the body of the message. Neither choice is completely satisfactory — deleting header lines causes the loss of valuable information but inserting header lines into the message text interferes with the proper operation of VMS MAIL’s EXTRACT/NOHEADER command. PMDF, unless configured otherwise by the system manager, uses the latter approach: PMDF preserves additional header lines by merging them into the text of the message.

Now, the addresses which appear in the VMS MAIL From:, To:, and Cc: header lines are not necessarily the same as those which appear in the original message’s header. The reason for this is that RFC 822 headers can specify several different types of From:, To:, and Cc: addresses. PMDF has to choose the addresses which best suit VMS MAIL’s usage of the VMS MAIL From:, To:, and Cc: header lines. The selection criteria used by PMDF are described below. Regardless of the addresses chosen, each address will be converted from RFC 822 format to VMS MAIL format (*e.g.,* IN%"address") before being given to VMS MAIL. Doing this allows REPLY commands in VMS MAIL to work correctly. (Also, “reply-to-all” type commands such as those in the Pathworks MAIL programs, will work correctly too.)

From:

PMDF tries to use the header line from the original message header that is most likely to contain the address to which replies should be directed. Whether or not PMDF uses any Resent- headers in this process is controlled by the `useresent` channel keyword. If the local channel has `useresent 2` or `useresent 1`, then the header lines that are used, in order of decreasing precedence, are:†

1. Resent-Reply-To: (if present)
2. Resent-From: (if present)
3. Reply-To: (if present)
4. From: (if present)
5. Resent-Sender: (if present)
6. Sender: (if present)
7. envelope return address (if non-empty)
8. MISSING_ADDRESS PMDF option value†

If the local channel has `useresent 0`, the default, then the header lines that are used, in order of decreasing precedence, are:

1. Reply-To: (if present)
2. From: (if present)
3. Sender: (if present)
4. envelope return address (if non-empty)
5. MISSING_ADDRESS PMDF option value†

To:

PMDF tries to use the header line from the original message header that is most likely to indicate to whom this version of the message was sent. If the local channel has `useresent 2`, then the fields that are used, in order of decreasing precedence, are:

† See Section 7.3.11 for a discussion of MISSING_ADDRESS.

1. Resent-To: (if present)
2. To: (if present)

If the local channel has `useresent 1` or `useresent 0`, the default, then the To: field is used.

If neither a Resent-To: or To: header line is present, then the VMS MAIL To: line will be left blank.

Cc:
PMDF tries to use the field from the message header that is most likely to indicate who the most recent Cc: recipients of the message were. If the local channel has `useresent 2`, then the fields that are used, in order of decreasing precedence, are:

1. Resent-Cc: (if present)
2. Cc: (if present)

If the local channel has `useresent 1` or `useresent 0`, then the Cc: field is used.

If neither a Resent-Cc: or Cc: header line is present, then the VMS MAIL Cc: line will be left blank.

Subject:
The Subject: line for VMS MAIL is simply copied verbatim out of the original message header. If no Subject: line is present, the VMS MAIL Subject: line is left blank.

19.3 Delivery and Read Receipts

PMDF provides support for delivery receipts (*i.e*, a confirmation message sent to you when your message reaches the recipient's mailbox) and limited support for read receipts (*i.e.*, a confirmation message sent to you when your message is actually read by its recipient). Please note that most, if not all, mail user agents which support read receipts allow the recipient to block them. This is typically the default; *i.e.,* most mail reading programs will not generate read receipts unless the reader of the message expressly approves them. Many people feel that read receipts are an invasion of privacy and should only be honored if the recipient chooses to do so.

Note that there are two separate issues concerning delivery and read receipts. First, a mechanism must exist with which to request delivery or read receipts for any given message. This support must exist on the system where the mail message originates. PMDF supports the generation of delivery and read receipt requests. [1]

Second, the support must exist at the receiving end to recognize and generate delivery or read receipts. PMDF provides the necessary support to honor delivery receipts (*i.e.,* emit a confirmation message when a message with a delivery receipt request is received). Many other mailers, notably "sendmail" based systems, also support delivery

[1] While not the topic of this chapter, note that PMDF-LAN also support receipt requests.

receipts. Not all mailers support them, however, so it is quite normal for receipt requests to be ignored when sent to non-PMDF mailers (or PMDF versions prior to V4.0).

It is the responsibility of the mail user agent to which a message is delivered to generate a read receipt when the recipient reads a message. This is not within PMDF's jurisdiction nor is it possible for PMDF to intervene. The best that PMDF can do is to demote read receipt requests to delivery receipt requests when delivering mail to a mail user agent, such as VMS MAIL, which PMDF knows cannot recognize, let alone act upon, read receipt requests. PMDF will pass read receipt requests to Gold-Mail, which is capable of acting upon read receipt requests by generating read receipts.

19.3.1 Requesting Delivery or Read Receipts

You can request a delivery receipt for a specific recipient listed in a single IN% construct by including the string

```
(delivery-receipt)
```

in the construct adjacent to the address. (Strings enclosed in parentheses are RFC 822 comments and won't otherwise affect the address.) For example, you might use an address of the form

```
IN%"postmaster@ymir.claremont.edu (delivery-receipt)"
```

to request delivery confirmation on a message to postmaster@ymir.claremont.edu.

This string will, by default, cause a delivery receipt request to be inserted into the message header of all copies of the message sent to recipients listed in that IN% construct. Other recipients will not be affected. The exact form of the delivery receipt request depends on the destination of the message. Delivery receipt requests appear as a Delivery-receipt-to: header in messages PMDF delivers to VMS MAIL mailboxes. Alternatively, PMDF can be configured to generate NOTARY (non-header) delivery receipt requests rather than any header requesting a delivery receipt; see Section 19.3.2.

The string

```
(read-receipt)
```

requests a read receipt. It operates in a similar fashion to the delivery receipt mechanism described above. PMDF will convert any read receipt requests it receives into delivery receipt requests when it knows that the mail user agent to which the mail is being delivered cannot honor a read receipt request (when delivering to VMS MAIL, for example). Read receipt requests appear as a Disposition-notification-to: header in messages PMDF delivers to VMS MAIL mailboxes.[2]

Note: The phrases `delivery-receipt` and `read-receipt` appearing within the parentheses are actually configurable via settings in the PMDF option file. The system administrator responsible for configuring PMDF determines the phrases used at a given site and can elect to change them from the defaults described here.

[2] The Disposition-notification-to: header is defined in RFC 2298. Versions of PMDF prior to PMDF V5.2 used a Read-receipt-to: header.

In the PMDF MAIL user agent, the /DELIVERY_RECEIPT and /READ_RECEIPT qualifiers of the SEND, FORWARD, and REPLY commands provide an alternate way to request receipts on a per-message basis.

Receipts are sent to the message originator by default. You can request that receipts for messages you originate be sent to a different address by defining the PMDF_DELIVERY_RECEIPT_TO logical to translate to the RFC 822 address to which you prefer that delivery receipts be sent. If you define this logical, receipts will be requested by default (no need to specify anything special in the address). You can then suppress the generation of delivery receipt requests by adding a

```
(no-delivery-receipt)
```

comment in the IN% construct. Similarly, the PMDF_READ_RECEIPT_TO logical can be used to specify the address to which read receipts should be sent. If set, it also causes read receipt requests to be generated by default, and the comment

```
(no-read-receipt)
```

can be used to suppress the generation of read receipt requests.

Note: The use of the PMDF_DELIVERY_RECEIPT_TO and PMDF_READ_RECEIPT_TO logicals is deprecated, since they unconditionally generate old style header receipt requests, rather than the newer sorts of requests in use on the Internet. That is, PMDF_DELIVERY_RECEIPT_TO unconditionally results in use of a Delivery-receipt-to: header, rather than a non-header NOTARY delivery receipt requestion, while PMDF_READ_RECEIPT_TO results in a Read-receipt-to: header, rather than the newer Disposition-notification-to: header.

Warning: Be very circumspect in the use of the PMDF_DELIVERY_RECEIPT_TO and PMDF_READ_RECEIPT_TO logicals. In general, you do not want to leave them constantly defined (unless they are defined to be "<>". At issue here is the danger of accidentally posting mail with an attached receipt request to a large mailing list. While many mailing lists will properly block such requests, quite a few do not and instead pass the request on to each and every list member. In such a case, as many as one receipt per list member will be generated (some addressees can be using mailers which ignore receipt requests). For a large mailing list with thousands of members, this means that your system can be *flooded* with mail.

19.3.2 Delivery Receipt Mechanisms: Header *vs.* NOTARY

PMDF actually supports two separate and distinct delivery receipt request mechanisms: envelope level delivery receipt requests as defined by RFC's 1891–1894 (often referred to as NOTARY), and the ad-hoc header style delivery receipt requests that were all that existed prior to NOTARY. (PMDF layered products such as PMDF-LAN also support "foreign" delivery receipt mechanisms, if they exist, to the extent possible.)

Newer mailers can support the NOTARY delivery receipt mechanism; support for the NOTARY mechanism can be expected to grow. Older mailers, however, such as older sendmail implementations, can not yet support NOTARY and the only hope of getting back delivery receipts from such mailers can be to send the delivery receipt requests in the old header style.

The channel keywords `reportheader`, `reportnotary`, `reportboth`, and `report-suppress`) when used on the L channel control whether the delivery receipt requests resulting from a `(delivery-receipt)` comment (or use of PMDF MAIL's /DELIVERY_RECEIPT qualifier) are header style requests, NOTARY style requests, or both, or whether no receipt requests are generated at all. `reportheader` is the default. To maximize the chances of receiving back requested delivery reports, whether the remote receiving side supports the header style request mechanism or NOTARY, one can set `reportboth`; however, note that if a message with both style requests is delivered via a mailer (such as PMDF) that supports both styles, then *two* delivery reports can be generated, one for each form of request.

19.4 Extensions to RFC 822

PMDF addresses can use certain non-standard, but sometimes useful, formats. These formats are not part of RFC 822 and, as such, constitute non-standard extensions to RFC 822. These extensions are:

- If an address does not contain an at sign, @, and hence does not contain the name of a destination system, then the name of the local host is used for the destination system.[3] Note that the elimination of the at sign can in turn eliminate the need for double quotes. For example, the address

  ```
  IN%user
  ```

  is interpreted as

  ```
  IN%"USER@local-host"
  ```

  where *local-host* is the local host name.[4]

  As an example, try sending mail to yourself, and, when you receive the message, look at the From: address. For instance, if the local host name is example.com and your username is mrochek, then the address specification in%mrochek will be interpreted as in%"MROCHEK@example.com".

  Forms such as

  ```
  IN%"user1, user2, user3"
  ```

  are also allowed. (*user1*, *user2*, and *user3* are not subject to separate logical name translation in this case.) Such an address specification is equivalent to

  ```
  IN%"user1@local-host, user2@local-host, user3@local-host"
  ```

[3] If PMDF is being invoked locally, the name of the local host is the official host name associated with PMDF's local channel on that system. In the case of remote usage via MAIL-11 over DECnet, the local host name is the name of the remote DECnet host using PMDF.

[4] Note that these addresses are not exactly equivalent: VMS MAIL and DECwindows MAIL will translate *user* as a logical name prior to passing it to PMDF. Adding double quotes will not prevent this translation from happening. The *@local-host* clause blocks such translations; explicit specification of it can be needed for this reason.

- Source-routed addresses without surrounding angle brackets are accepted even though RFC 822 does not formally allow such syntax. For example,

  ```
  IN%"@cunyvm.cuny.edu:fresnel@kitvax"
  ```

  is treated as

  ```
  IN%"<@cunyvm.cuny.edu:fresnel@kitvax>"
  ```

- Personal names can contain periods and other punctuation characters without being surrounded by double quotes, even though the quotes are called for by RFC 822. For example,

  ```
  IN%"Augustin J. Fresnel <fresnel@kitty.farm.org>"
  ```

  is treated as

  ```
  IN%"'Augustin J. Fresnel' <fresnel@kitty.farm.org>"
  ```

- IBM NOTES format addresses are accepted. These are addresses of the form

  IN%"*user* AT *host*"

  and are converted to

  IN%"*user*@*host*"

- Addresses of the form

  IN%"*user*@*host1*@*host2*"

  are converted to

  IN%"*user*%*host1*@*host2*"

- Addresses of the form

  IN%"@*host*:*user*"

  are converted to

  IN%"@*host*:*user*@*localhost*"

- So-called "DECnet-style" addresses are accepted and converted into RFC 822 format. That is, addresses of the form

  IN%*system*::*user*

  are converted to

  IN%"*USER*@*SYSTEM*"

  Note the forced conversion to upper case. This conversion is done by VMS MAIL and cannot be eliminated or undone. This address format is discouraged for this reason; RFC 822 mandates preservation of case in the local part (the part to the left of the at sign) of all addresses. In addition, *system* is translated as a logical name. Multiple routing systems can be specified; *e.g.,* an address such as

IN%*system1*::*system2*::*system3*::*user*

is converted to

IN%"*USER%SYSTEM3%SYSTEM2@SYSTEM1*"

Note: Surrounding double quotes *can not be used* with DECnet-style addresses. This restriction is imposed by VMS MAIL, not PMDF.

In all cases PMDF reformats the addresses to comply with RFC 822, so no illegal addresses are passed to other mail systems.

19.5 Obtaining Headers from Message Text

Some applications, especially those forwarding or replying using existing message text, can provide additional message headers as plain text within the body of the message. The special logical PMDF_RELAYING can be used to instruct PMDF to read additional headers from the top of the message's body.

PMDF_RELAYING is also used to construct an additional Received: header line. The translation of PMDF_RELAYING is placed on the Received: line along with the current date and time. The result should conform to RFC 822 conventions for Received: lines; PMDF does not check to insure that this is the case, however.

The additional Received: header will not be generated if PMDF_RELAYING translates to a single space character.

Nonprivileged use of PMDF_RELAYING presents a security hole — users could use this mechanism to forge message headers. For this reason PMDF_RELAYING must be defined as an executive mode logical in order to have any effect. This precludes its use by nonprivileged users, yet makes it available to the system-level applications that might actually need it.

Under no circumstances should PMDF_RELAYING be defined system-wide — it will interfere with the use of PMDF by normal users.

19.6 The DELIVER Message Delivery System

The DELIVER facility permits users to perform automatic filtering on their incoming messages based on the presence of arbitrary substrings in the From:, Subject:, or other header lines, *e.g.*, forward certain messages, direct certain messages to particular VMS MAIL folders, direct certain messages straight to files, generate automatic replies, execute a DCL command upon receipt of certain messages, *etc.*; a complete description of the use of this facility can be found in the *PMDF User's Guide*.

When PMDF is initially installed, the DELIVER facility is enabled: a user triggers the DELIVER facility for mail handled by PMDF by creating an appropriate `mail.delivery` file in their own default login directory. Any necessary processing is performed by batch jobs; if a user does not specify a batch queue, then the default

DELIVER batch queue specified during PMDF installation (pointed to by the DELIVER_BATCH logical name) will be used. The system manager can unconditionally disable the use of DELIVER via the PMDF option USE_MAIL_DELIVERY or can choose a different name for DELIVER files via the PMDF option MAIL_DELIVERY_FILENAME; see Section 7.2.

20 DECnet Channels (OpenVMS and Tru64 UNIX)

Note: Presently, DECnet channels are only supported on OpenVMS systems.

PMDF includes an SMTP over DECnet channel which uses the standard SMTP protocol to convey messages, over DECnet as a transport agent. On OpenVMS, PMDF also includes two additional types of channels which use DECnet as a transport agent. That is, these three types of channels are:

- *SMTP over DECnet channels* which use the SMTP protocol and task-to-task DECNET;
- (OpenVMS only) *PhoneNet over DECnet channels* which use the PhoneNet protocol and task-to-task DECnet; and
- (OpenVMS only) *DECnet MAIL-11 channels* which use the DECnet based MAIL-11 protocol.

VMS

The last of the three types of channels available on OpenVMS, DECnet MAIL-11 channels, is an obsolete type of channel provided for compatability with older versions of PMDF and for communicating with other OpenVMS systems which do not have PMDF. Whenever possible, either of the first two types of channels, which use direct DECnet links to form a transport layer for SMTP or PhoneNet message transfers, should be used instead.

Note that the terminology used here is a bit confusing. This is brought about from the fact that DECnet is both a transport system and a mail system. *SMTP over DECnet channels*, or alternatively on OpenVMS, *PhoneNet over DECnet channels*, are used to link multiple systems running PMDF that happen to be attached to the same DECnet. They are entirely distinct from the *DECnet MAIL-11 channels*. This document always refers to DECnet's own message delivery system as DECnet MAIL-11 and to DECnet's transport mechanism as simply DECnet.

20.1 SMTP Over DECnet Channels

SMTP over DECnet channels are used to link systems running PMDF that are also connected via DECnet. These channels are comparable to the PhoneNet over DECnet channels described in the following section; the difference is that these channels use standardized SMTP protocols instead of the nonstandardized PhoneNet protocol. These channels can be used to communicate with systems that can speak SMTP over DECnet, notably other OpenVMS systems running PMDF.

SMTP over DECnet channels use unidirectional master and slave programs. The master program runs when PMDF has outgoing messages to send. The slave program runs in response to incoming DECnet requests.

On OpenVMS, DECnet SMTP channels can be generated by the automatic configuration generator.

20.1.1 Setting Up the Channel

Both PMDF and DECnet must be installed on both systems before a SMTP over DECnet channel can be set up. Once this is done, activating a SMTP over DECnet channel between them is quite straightforward as below.

20.1.1.1 Installing PMDF as a Known Object

PMDF must be installed as a known DECnet object on each system which is going to use a SMTP over DECnet channel.

First check what DECnet object numbers are currently in use and decide upon an unused one to assign to PMDF. Object numbers from 128 to 255 are available for customer use. The actual number used is irrelevant provided it is not used for any other purpose and all systems agree on the same number. Use the NCL command

```
NCL> SHOW SESSION CONTROL APPLICATION * ALL CHARACTERISTICS
```

or the NCP command

```
NCP> LIST KNOWN OBJECTS
```

to get a list of currently defined objects and their associated object numbers. Then choose an unused DECnet object number for PMDF's use.

VMS

On OpenVMS, you will also need to know the username of the PMDF account (usually just PMDF), to specify where *pmdf_account* is shown below. On a DECnet Phase IV system, you will additionally need to know the account's password, to specify where *pmdf_password* is shown below. These must match the local PMDF account for proper operation — if the username or password for the PMDF account is changed, the DECnet database will have to be updated as well.

On a DECnet/OSI system, the DECnet object installation is performed using NCL.

VMS

On OpenVMS, use NCL as follows:

```
$ RUN SYS$SYSTEM:NCL
NCL> CREATE SESSION CONTROL APPLICATION PMDFSMTP
```

```
NCL> SET SESSION CONTROL APPLICATION PMDFSMTP -
       ADDRESSES = {NAME = PMDFSMTP, NUMBER = xxx }, -
       OUTGOING PROXY = FALSE, -
       INCOMING PROXY = FALSE, -
       NODE SYNONYM = TRUE, -
       IMAGE NAME = PMDF_COM:dsmtp_slave.com, -
       USER NAME = "pmdf_account"

# ncl
ncl> create session control application pmdfsmtp
ncl> set session control application pmdfsmtp -
       addresses = {name = pmdfsmtp, number = xxx }, -
       outgoing proxy = false, -
       incoming proxy = false, -
       node synonym = true, -
       image name = "/pmdf/bin/dsmtp_slave", -
       user name = "pmdf"
```

Note the use of quotes to preserve lowercase, when case matters.

VMS

On a DECnet Phase IV OpenVMS system, the DECnet object installation is performed using the Network Control Program (NCP), as follows:

```
$ RUN SYS$SYSTEM:NCP
NCP> DEFINE OBJECT PMDFSMTP -
            NUMBER xxx -
            FILE PMDF_COM:dsmtp_slave.com -
            USER pmdf_account -
            PASSWORD pmdf_password -
            ACCOUNT SYSTEM -
            PROXY NONE
NCP> SET    OBJECT PMDFSMTP -
            NUMBER xxx -
            FILE PMDF_COM:dsmtp_slave.com -
            USER pmdf_account -
            PASSWORD pmdf_password -
            ACCOUNT SYSTEM -
            PROXY NONE
```

The first command defines the PMDF object in the permanent DECnet database and the second defines the PMDF object in the volatile DECnet database.

You can have outgoing connections made by DSMTP channels identify themselves with the DECnet cluster alias instead of using the specific DECnet node name they happen to be running on. This is enabled by adding the NCL clause “OUTGOING ALIAS = TRUE” or the NCP clause “ALIAS OUTGOING ENABLED” to the commands shown above.

20.1.1.2 Adding the Channel to the Configuration File

The last step in setting up a SMTP over DECnet channel is to add the channels to the appropriate configuration files.

Note: The PMDF configuration utility, PMDF CONFIGURE, can be used to configure your system automatically with the appropriate DSMTP channels. When the PMDF configuration utility is used, all of the necessary configuration steps except for those described in Section 20.1.1.1 are done for you automatically. Consult the appropriate edition of the *PMDF Installation Guide* for instructions on using the configuration utility.)

A typical channel table entry for an SMTP over DECnet channel is of the form shown in Example 20–1; the corresponding rewrite rules are shown in Example 20–2.

Example 20–1 Sample SMTP Over DECnet Channel Block

```
dsmtp_local single_sys smtp
DSMTP-DAEMON
ditmelb.oz.au     ditmb
praxa.com.au      praxa
```

Note that while any channel name beginning with "dsmtp_" can be used with the master program (*i.e.,* to send mail via a SMTP over DECnet link), the only channel that will be used with the slave program (*i.e.,* to receive mail over a SMTP over DECnet link) is dsmtp_local.

Example 20–2 Rewrite Rules for Example 20–1

```
praxa.com.au      $u@praxa.com.au
ditmelb.oz.au     $u@ditmelb.oz.au
```

Here access is provided to two systems, ditmelb.oz.au and praxa.com.au. Their corresponding DECnet node names are ditmb and praxa. This format is analogous to that used in setting up the DECnet MAIL channel (channel d). As many systems as desired can be listed in the same channel block.

The `daemon router` keyword clause can be used to set up a point to point connection to a gateway through a SMTP over DECnet connection. See the documentation on TCP/IP gateway channels in Chapter 21 for additional information on how to set up a gateway channel.

After the configuration files are set up the channel should be ready for use.

20.1.1.3 Channel Option File

While it is extremely rare that controlling SMTP characteristics of an SMTP over DECnet channel is of interest, an option file can be used to control various such SMTP characteristics. See the list of options in Section 21.1.2.2; most of those options (those relating to the SMTP protocol itself, rather than specifically to TCP/IP transport issues) can also be applied to SMTP over DECnet channels.

Such an option file must be named `x_option` where *x* is the name of the channel, and stored in the PMDF table directory. Since the name of the channel is usually dsmtp_local, the option file is usually `PMDF_TABLE:dsmtp_local_option.` .

20.1.2 Network Service Logs

Once a slave channel program is successfully started, it will log eny errors or debugging output, to a normal PMDF channel log file, *e.g.*, `PMDF_LOG:dsmtp_local_slave.log` .

VMS

On OpenVMS, however, since slave operations are initiated by DECnet directly, DECnet itself creates its own log files for slave operations. These log files appear in the default directory of the account under which the DECnet PMDFSMTP object runs; that account is normally the PMDF account hence normally the DECnet log files are in `PMDF_ROOT:[log]` on OpenVMS, and have the name `netserver.log`. Note that unlike other PMDF log files, these DECnet channel slave log files created directly by DECnet are not named according to the channel that created them.

20.2 PhoneNet Over DECnet Channels (OpenVMS)

PhoneNet over DECnet channels can be used to link systems running PMDF that are also connected via DECnet. The PhoneNet over DECnet channel software uses PhoneNet's phone protocol on top of transparent DECnet links. It is *not* compatible with MAIL-11 (the protocol used by VMS MAIL to communicate with other VMS MAIL systems on DECnet). PhoneNet over DECnet channel names always begin with dn_ in the PMDF configuration file.

DECnet channels use bidirectional master and slave programs. The master program runs when PMDF has outgoing messages to send. The slave program runs in response to incoming DECnet task requests. As these channel programs are bidirectional, note that both the master and slave programs can deliver messages in both directions; or in other words, when the channel runs initiated in either the master or slave direction, it also "polls" for messages travelling in the other direction.

PhoneNet over DECnet channels are generated by the PMDF configuration utility, if any are needed.

20.2.1 Setting Up the Channel

Both PMDF and DECnet must be installed on both systems before a PhoneNet over DECnet channel can be set up. Once this is done, activating a PhoneNet over DECnet channel between them is quite straightforward as described in the following sections.

The PMDF configuration utility on OpenVMS, PMDF CONFIGURE, can be used to automatically configure your system with the appropriate PhoneNet over DECnet (DN) channels. When a PMDF configuration utility sets up these channels for you, you only need to define the DECnet object as discussed in Section 20.2.1.1; you do not need to do anything else. *Process Software recommends that a PMDF configuration utility be used to generate PhoneNet over DECnet channels.* The configuration utility is documented in the OpenVMS edition of the *PMDF Installation Guide*.

20.2.1.1 Installing PMDF as a Known Object

PMDF must be installed as a known DECnet object on each system which is going to use a PhoneNet over DECnet channel. Run the Network Control Program (NCP) and enter the following commands:

```
$ RUN SYS$SYSTEM:NCP
NCP> DEFINE OBJECT PMDF -
             NUMBER xxx -
             FILE PMDF_COM:dn_slave.com -
             USER pmdf_account -
             PASSWORD pmdf_password -
             ACCOUNT SYSTEM -
             PROXY NONE
NCP> SET    OBJECT PMDF -
             NUMBER xxx -
             FILE PMDF_COM:dn_slave.com -
             USER pmdf_account -
             PASSWORD pmdf_password -
             ACCOUNT SYSTEM -
             PROXY NONE
```

The first command defines the PMDF object in the permanent DECnet database and the second defines the PMDF object in the volatile DECnet database.

If you are using DECnet/OSI, the equivalent NCL commands are as follows:

```
$ RUN SYS$SYSTEM:NCL
NCL> CREATE SESSION CONTROL APPLICATION PMDF
NCL> SET SESSION CONTROL APPLICATION PMDF -
       ADDRESSES = {NAME = PMDF, NUMBER = xxx }, -
       OUTGOING PROXY = FALSE, -
       INCOMING PROXY = FALSE, -
       NODE SYNONYM = TRUE, -
       IMAGE NAME = PMDF_COM:dn_slave.com, -
       USER NAME = "pmdf_account"
```

xxx should be an unused DECnet object number. Object numbers from 128 to 255 are available for customer use. The actual number used is irrelevant provided it is not used for any other purpose and all systems agree on the same number. Use the LIST KNOWN OBJECTS command in NCP (SHOW SESSION CONTROL APPLICATION * ALL CHARACTERISTICS in NCL) to get a list of currently defined objects and their associated numbers.

pmdf_account should be the username associated with the PMDF account (usually just PMDF) and *pmdf_password* should be the password for the PMDF account. These must match the local PMDF account for proper operation — if the username or password for the PMDF account is changed, the DECnet database will have to be updated as well.

You can have outgoing connections made by DN channels identify themselves with the DECnet cluster alias instead of using the specific DECnet node name they happen to be running on. This is enabled by adding the clause "ALIAS OUTGOING ENABLED" to the NCP commands shown above. The NCL equivalent is "OUTGOING ALIAS = TRUE".

20.2.1.2 Adding the Channel to the Configuration File

The last step in setting up a PhoneNet over DECnet channel is to add the channels to the appropriate configuration files. You do not need to do this if your PhoneNet over DECnet channels were created with the configuration utility; see the recommendation in Section 20.2.1.

PhoneNet over DECnet channel names have a rigid syntax; they are always of the form "dn_*remotenode*", where *remotenode* is the name of the system to which the channel connects. This means that a single channel has different names on each end; *e.g.*, a channel from DECnet node A to node B is called dn_b on node A and dn_a on node B.

A separate channel is required for each link to a remote system. Connection requests from systems not in the channel table will be refused.

Finally, the rewrite rules in the configuration files should be edited to refer to the new channels as appropriate. See the example below for additional information.

After the configuration files are set up, the channel should be ready for use.

20.2.2 Example Configuration

Suppose that two systems, ALPHA and BETA, are both running PMDF and are attached to the same DECnet. ALPHA is registered as ALPHA.DOOF.COM in the Internet world and is the server for the DOOF subdomain. ALPHA has a PhoneNet (dial-up) link to CSNET and BETA has a PhoneNet (dial-up) link to BITNET. The BITNET name for BETA is BETA.BITNET, although the Internet name BETA.DOOF.COM is also supported.

The configuration should then route all BITNET traffic to BETA and all other network traffic to ALPHA. A PhoneNet over DECnet channel will be used to transfer mail from ALPHA to BETA. Possible configuration files for nodes ALPHA and BETA are shown in Examples 20–3 and 20–4 below.

Example 20–3 PhoneNet Over DECnet Channel Configuration for Node ALPHA

```
! ALPHA.CNF - Configuration file for hypothetical node ALPHA
! Written by Ned Freed, 20-Jul-1986.
!
! Handle local systems routing first
!
ALPHA           $U@ALPHA.DOOF.COM
ALPHA.DOOF.COM $U@ALPHA.DOOF.COM
BETA            $U@BETA.BITNET
BETA.DOOF.COM  $U@BETA.BITNET
BETA.BITNET    $U@BETA.BITNET
!
! Force match on any other systems in .DOOF.COM,
! since this is the server for the subdomain. This
! prevents message loops on bad addresses.
!
.DOOF.COM     $U@$H.DOOF.COM
!
! BITNET messages are routed to BETA
!
.BITNET       $U%$H.BITNET@BETA.BITNET
!
! Other top level domains - only a few to give the flavor
!
.AR           $U%$H$D@relay.cs.net
.ARPA         $U%$H$D@relay.cs.net
.AT           $U%$H$D@relay.cs.net
.AU           $U%$H$D@relay.cs.net
.BE           $U%$H$D@relay.cs.net

l
ALPHA.DOOF.COM

! PhoneNet (dial-up) channel for connecting with CSNET
p network
RELAY.CS.NET

! PhoneNet over DECnet channel for connecting with BETA
dn_beta network
BETA.BITNET
```

Example 20–4 PhoneNet Over DECnet Channel Configuration for Node BETA

Example 20–4 Cont'd on next page

Example 20–4 (Cont.) PhoneNet Over DECnet Channel Configuration for Node BETA

```
! BETA.CNF - Configuration file for hypothetical node BETA
! Written by Ned Freed, 20-Jul-1986.
!
! Handle local systems routing first
!
BETA            $U@BETA.BITNET
BETA.DOOF.COM   $U@BETA.BITNET
BETA.BITNET     $U@BETA.BITNET
ALPHA           $U@ALPHA.DOOF.COM
ALPHA.DOOF.COM $U@ALPHA.DOOF.COM
!
! BITNET messages are queued to the BITNET channel
!
.BITNET       $U%$H.BITNET@RELAY.BITNET
!
! Other top level domains go to ALPHA - only a few shown
!
.AR           $U%$H$D@ALPHA.DOOF.COM
.ARPA         $U%$H$D@ALPHA.DOOF.COM
.AT           $U%$H$D@ALPHA.DOOF.COM
.AU           $U%$H$D@ALPHA.DOOF.COM
.BE           $U%$H$D@ALPHA.DOOF.COM

l 822
BETA.BITNET

! PhoneNet (dial-up) channel for connecting with BITNET
p 733 network
RELAY.BITNET

! PhoneNet over DECnet channel for connecting with ALPHA
dn_alpha network
ALPHA.DOOF.COM
```

20.2.3 Network Service Logs

In addition to any normal PMDF slave log files that can be created for a PhoneNet over DECnet channel, since slave operations are initiated by DECnet directly, DECnet itself creates log files for slave operations. Such additional log files appear in the default direction of the account under when the DECnet PMDF object runs; that account is normally the PMDF account, hence normally the DECnet log files are in `PMDF_ROOT:[LOG]` on OpenVMS, and have the name `netserver.log`. Note that unlike normal PMDF log files, *e.g.*, unlike any `dn_`*`remotenode`*`_slave.log` log files which can also be created, the DECnet channel slave log files created directly by DECnet are not named according to the channel that created them.

20.3 DECnet MAIL-11 Channels (OpenVMS)

Versions of PMDF prior to version 3.2 used the local channel to deliver messages across DECnet using VMS MAIL. While this can still be done, the use of channel DECnet MAIL-11 channels (d or d_ channels) is preferred for this operation. These channels are described more fully in Chapter 18; in particular, see Section 18.2

Note: PMDF also includes two true DECnet channels, which use direct DECnet links to form a transport layer for PhoneNet or SMTP message transfers. DECnet channels are used to link multiple systems running PMDF that happen to be attached to the same DECnet. They are entirely distinct from DECnet MAIL-11 channels. The terminology is quite confusing and is due to the fact that DECnet is both a transport mechanism and mail system in its own right, yet the same name is used for both functions. This document always refers to DECnet's own message delivery system as DECnet MAIL-11 and to DECnet's transport mechanism as simply DECnet.

The automatic configuration generator generates a DECnet MAIL-11 channel if one is needed.

21 TCP/IP Channels

TCP/IP channels are used to link PMDF to TCP/IP based networks such as the Internet. The TCP/IP channels all use the Simple Mail Transfer Protocol (SMTP), including extensions such as those described in RFCs 1426, 1869, 1870, and 1891.[1] The description of this protocol and extensions to it can be found in files such as `rfc1426.txt`, `rfc1869.txt`, `rfc1870.txt`, `rfc2821.txt`, and `rfc2822.txt` in the `rfc` subdirectory of the PMDF documentation directory.

On UNIX and NT, PMDF's multithreaded TCP SMTP channel uses the TCP/IP network protocol stack supplied with the operating system.

On OpenVMS, PMDF supports the following TCP/IP packages:

- Hewlett-Packard TCP/IP Services [a.k.a. UCX]
- MultiNet (Process Software)
- TCPware (Process Software)

On OpenVMS, the multithreaded TCP SMTP channel uses DECthreads in order to handle multiple, simultaneous SMTP sessions in a single process. [2] The multithreaded TCP SMTP channel thus requires OpenVMS V6.1 or later, Pascal RTL V5.0-15 (VAX) or V5.0-18 (Alpha) or V5.0-25 (I64) or later, and as the underlying TCP/IP package, any one of MultiNet V3.5A or later with all UCXDRIVER patches (upgrading to the latest MultiNet is recommended), Pathway V2.5.x with all patches (including C82195 as of this printing) or later, TCPware V5.0 or later[3], or TCP/IP Services for OpenVMS V4.0 or later (with UCX V4.1 requiring upgrading to at least ECO 5). Contact Process Software or your PMDF distributor if you have any questions regarding version compatibility with these TCP/IP packages.

The multithreaded TCP SMTP channel includes a multithreaded SMTP server which runs under the control of the PMDF Service Dispatcher. Outgoing SMTP mail is processed by the multithreaded PMDF channel program (`tcp_smtp_client`), run as needed under the control of `master.com` (OpenVMS) or the PMDF Job Controller (UNIX and NT).

Configuration instructions are presented in this chapter for the multithreaded TCP/IP channel.

[1] Two additional SMTP channels, SMTP over task-to-task DECnet and "generic" SMTP channels, are presented in Chapter 20 and Chapter 26.

[2] The multithreaded TCP SMTP channel replaces PMDF single-threaded TCP/IP channels specific to the above TCP/IP packages. For sites upgrading from older versions of PMDF, see the OpenVMS Edition of the *PMDF Installation Guide* for instructions on switching to using the multithreaded TCP SMTP channel.

[3] PMDF has been tested against TCPware versions through 5.5.

Section 21.3, towards the end of this chapter describes how to establish a "gateway" channel to route mail through another system, as for instance to a mailhub or firewall system. Section 21.4, at the end of this chapter, describes performing "polling" with TCP/IP channels, that is, requesting that a remote system attempt to deliver any stored messages to your system; for instance, this can be particularly useful over "intermittent" sorts of TCP/IP links, such as dial-up connections.

21.1 Setting Up the Channel

Before configuring a TCP/IP channel, you must select a domain name (*e.g.,* "naples.example.com") for each of the machines which will run PMDF. Associated with each domain name is an IP address (*e.g.,* `192.168.1.1`). If the implementation of TCP/IP which you use supports domain literals and you want to configure PMDF to use domain literal addressing (*e.g.,* `bob@[192.168.1.1]`), then you will also want to have at hand the IP addresses for each of the machines for which PMDF will be configured.

Note: The `pmdf configuration` utility will create TCP/IP channels for you. If you use this utility to configure TCP/IP channels, then many configuration steps will have been performed for you. On OpenVMS you must still be sure to disable the native SMTP server, as described in Section 21.1.3. On all platforms, be sure to complete all the configuration utility checklist steps, including configuring the PMDF Service Dispatcher. *Process Software strongly recommends that the configuration utility be used to configure TCP/IP channels.* Documentation on the configuration utility can be found in the appropriate edition of the *PMDF Installation Guide*.

21.1.1 Configuring the TCP SMTP Channel

If you do not have a TCP/IP channel in your configuration you can create one by adding a channel block to the PMDF configuration file that looks like this:

```
tcp_local single_sys smtp
TCP-DAEMON
```

The channel name must be `tcp_local` and the `single_sys` and `smtp` keywords are required. The `single_sys` keyword tells PMDF that only a single system is allowed in each message file since each message file will be associated with a single TCP connection. The `smtp` keyword activates the SMTP parser routines.

Rewrite rules need to be added to the configuration file to map system or domain names onto the `tcp_local` channel. If you used the PMDF configuration generator and told it that you wanted TCP/IP support, it should have already produced applicable rewrite rules. Note that since the single `tcp_local` channel can connect to many hosts, the channel host name is the pseudonym `TCP-DAEMON`. Rewrite rules should rewrite to the pseudonym, and not simply to the destination host. For example:

```
NODE.EXAMPLE.COM            $U%$D@TCP-DAEMON
```

The multithreaded TCP SMTP channel supports domain literal addressing. Internet Requirements (see RFC 1123) mandate that an Internet host be able to accept a domain literal specifying its own IP address. You should add a rewrite rule to your configuration file of the form

```
[1.2.3.4]        $U@official-local-host-name
```

where *1.2.3.4* is your IP address and *official-local-host-name* is the official host name on your local channel. If all other domain literals are to be targeted to the channel a rewrite rule of the form

```
[]               $U%[$L]@TCP-DAEMON
```

can also be appropriate.

If many systems accessible via TCP/IP are grouped in a couple of common domains, the use of more general rewrite rules should be considered. For example, suppose that a large number of systems in the `.example.com` domain are accessible via TCP/IP. Then the rewrite rule

```
.EXAMPLE.COM         $U%$H$D@TCP-DAEMON
```

would tell PMDF that any system in the `.EXAMPLE.COM` domain can be reached via TCP/IP. Exceptions (*e.g.,* systems in the `.EXAMPLE.COM` domain that are not reachable via TCP/IP) can be handled by inserting additional more specific rewrite rules.

The only disadvantage to this scheme is that errors like sending to a nonexistent system in the `.EXAMPLE.COM` domain will not be detected until PMDF actually attempts to deliver the message to the nonexistent system.

21.1.2 TCP/IP Channel Option Files

An option file can be used to control various characteristics of TCP/IP channels.[4] Such an option file must be named *x*`_option` where *x* is the name of the channel, and stored in the PMDF table directory. Since the name of the channel is usually `tcp_local`, the option file is usually `PMDF_TABLE:tcp_local_option.` on OpenVMS or `/pmdf/table/tcp_local_option` on UNIX or `C:\pmdf\table\tcp_local_option` on NT. Note that while master channel programs (the outgoing channel direction or SMTP client) read their option file each time they run, the slave channel program (the SMTP server) reads the option file only when it is first started, hence will not see changes until restarted.[5]

[4] Most of the options described actually relate to the SMTP protocol itself, rather than to the TCP/IP transport. As such, other PMDF channels that use the SMTP protocol over other transports can have similar options.

[5] Also note that for incoming messages, such TCP/IP *channel* options are actually options for the SMTP server itself; in particular, the only channel options that matter for incoming messages are those for the SMTP server's default channel, not any channel options for possible other channels that can putatively handle the incoming message due to a `switchchannel` channel keyword-based "switching".

21.1.2.1 Format of the Option File

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

option=value

value can be either a string or an integer, depending on the option's requirements. If the option accepts an integer value a base can be specified using notation of the form *b%v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

21.1.2.2 Available TCP/IP Channel Options

The available options are:

ALLOW_ETRNS_PER_SESSION (integer)
Set a limit on the number of ETRN commands accepted per connection. The default is 1. See also Section 2.3.4.34 for a discussion of channel keywords affecting PMDF's response to ETRN commands.

ALLOW_REJECTIONS_BEFORE_DEFERRAL (integer)
Set a limit on the number of bad RCPT TO: addresses that are allowed during a single connection. When this option is enabled, after the specified number of To: addresses have been rejected, all subsequent recipients, good or bad, are rejected with a temporary error.

ALLOW_RECIPIENTS_PER_TRANSACTION (integer)
Set a limit on the number of recipients allowed per message. The default is no limit.

ATTEMPT_TRANSACTIONS_PER_SESSION (integer)
Set a limit on the number of messages PMDF will attempt to transfer during any one connection session. The default is no limit.

BANNER_ADDITION (string)
Add the specified string to the SMTP banner line. The vertical bar character is not permitted in the string.

CHECK_SOURCE (0 or 1)
The PMDF SMTP server normally attempts to determine the name of the host from which a connection has been received, as specified by the ident* channel keywords discussed in Section 2.3.4.40. When the determined name does not match the name presented by the remote SMTP client on the HELO/EHLO line, the CHECK_SOURCE option controls whether the name found from a DNS lookup (or the IP domain literal, if DNS lookups have been disabled such as with the identnonenumeric channel keyword) is included in the constructed Received: header as a comment after the presented name. A value of 1 (the default) enables the inclusion of the determined name when different from the presented name; a value of 0 disables the inclusion of any such comment and thus eliminates one of the more useful checks of message validity.

COMMAND_RECEIVE_TIME (integer)
This option specifies, in minutes, how long to wait to receive general SMTP commands, (commands other than those with explicitly specified time out values set using other specifically named options). The default value is 10.

COMMAND_TRANSMIT_TIME (integer)
This option specifies, in minutes, how long to spend transmitting general SMTP commands, (commands other than those with explicitly specified time out values set using other specifically named options). The default value is 10.

DATA_RECEIVE_TIME (integer)
This option specifies, in minutes, how long to wait to receive data during an SMTP dialogue. The default is 5.

DATA_TRANSMIT_TIME (integer)
This option specifies, in minutes, how long to spend transmitting data during an SMTP dialogue. The default is 10.

DISABLE_ADDRESS (0 or 1)
The PMDF SMTP server implements a private command XADR. This command returns information about how an address is routed internally by PMDF as well as general channel information. Releasing such information can consistute a breach of security for some sites. Setting the DISABLE_ADDRESS option to 1 disables the XADR command. The default is 0, which enables the XADR command.

DISABLE_CIRCUIT (0 or 1)
The PMDF SMTP server implements a private command XCIR. This command returns PMDF circuit check information. Releasing such information can consistute a breach of security for some sites. Setting the DISABLE_CIRCUIT option to 1 disables the XCIR command; setting DISABLE_CIRCUIT to 0 enables the XCIR command. If DISABLE_CIRCUIT is not explicitly set, then use of this XCIR command is controlled by the DISABLE_GENERAL option setting.

DISABLE_EXPAND (0 or 1)
The SMTP EXPN command is used to expand mailing lists. Exposing the contents of mailing lists to outside scrutiny can constitue a breach of security for some sites. The DISABLE_EXPAND option, when set to 1, disables the EXPN command completely. The default value is 0, which causes the EXPN command to work normally.

Note that mailing list expansion can also be blocked on a list-by-list basis with the [EXPANDABLE] and [NONEXPANDABLE] named parameters.

DISABLE_GENERAL (0 or 1)
The PMDF SMTP server implements a private command XGEN. This command returns status information about whether a compiled configuration and compiled character set are in use. Releasing such information can constitute a breach of security for some sites. Setting the DISABLE_GENERAL option to 1 disables the XGEN command. The default is 0, which enables the XGEN command.

DISABLE_STATUS (0 or 1)
The PMDF SMTP server implements a private command XSTA. This command returns status information about the number of messages processed and currently in the PMDF channel queues. Releasing such information can consistute a breach of security for some sites. Settting the DISABLE_STATUS option to 1 disables the XSTA command. The default is 0, which enables the XSTA command.

DOT_TRANSMIT_TIME (integer)
This option specifies, in minutes, how long to spend transmitting the dot (period) terminating the data in an SMTP dialogue. The default is 10.

HIDE_VERIFY (0 or 1)

The SMTP VRFY command can be used to establish the legality of an address prior to actually using it. Unfortunately this command has been abused by automated query engines in some cases. The HIDE_VERIFY option, when set to 1, tells PMDF not to return any useful information in the VRFY command result. The default value is 0, which causes VRFY to act normally. See also the channel keywords controlling this behavior, described in Section 2.3.4.36.

LOG_BANNER (0 or 1)

The LOG_BANNER option controls whether the remote SMTP server banner line is included in mail.log* file entries when the logging channel keyword is enabled for the channel. A value of 1 (the default) enables logging of the remote SMTP server banner line; a value of 0 disables it. LOG_BANNER also affects whether a remote SMTP banner line, if available, is included in bounce messages generated by the channel.

LOG_CONNECTION (integer)

The LOG_CONNECTION option controls whether or not connection information, *e.g.*, the domain name of the SMTP client sending the message, is saved in mail.log file entries and the writing of connection records when the logging channel keyword is enabled for the channel. This value is a decimal integer representing a bit-encoded integer, the interpretation of which is given in the table below.

Bit	Value	Usage
0	1	When set, connection information is included in E and D log records.
1	2	When set, connection open/close/fail records are logged by message enqueue and dequeue agents such as the SMTP and X.400 clients and servers.
2	4	When set, I records are logged recording ETRN events.

Bit 0 is the least significant bit.

This channel option defaults to the setting of the global PMDF option LOG_CONNECTION as set in the PMDF option file; see Chapter 7. This channel option can be set explicitly to override on a per-channel basis the behavior requested by the global option.

LOG_TRANSPORTINFO (0 or 1)

The LOG_TRANSPORTINFO controls whether or not transport information, such as the sending and receiving side IP numbers and port numbers, are included in mail.log file entries when the logging channel keyword is enabled for the channel. A value of 1 enables transport information logging. A value of 0 disables it. This channel option defaults to the setting of the global PMDF option LOG_CONNECTION as set in the PMDF option file; see Chapter 7. It can be set explicitly for a channel to control the logging of transport information regardless of whether connection logging is enabled. LOG_TRANSPORTINFO also affects whether transport information, if available, is included in bounce messages generated by the channel.

LONG_LINE_MODE (0, 1, or 2)

This option specifies how PMDF should handle receiving lines that violate SMTP maximum length standards. A value of 0 (the default) tells PMDF to truncate such standards-violating long lines. A value of 1 tells PMDF to reject messages that contain long lines. A value of 2 tells PMDF to wrap the long lines at 1000 characters.

MAIL_TRANSMIT_TIME (integer)
This option specifies, in minutes, how long to spend transmitting the SMTP command MAIL FROM:. The default is 10.

MAX_CLIENT_THREADS (integer)
An integer number indicating the maximum number of simultaneous, outbound connections that the client channel program will allow. Note that multiple processes can be used for outbound connections, depending on how you have channel processing queues setup. This option controls the number of threads per process. The default value is 10.

MAX_MX_RECORDS (integer <= 32)
Specify the maximum number of *MX* records that PMDF should try using when attempting to deliver a message. The maximum value is 32, which is also the default.

RCPT_TRANSMIT_TIME (integer)
This option specifies, in minutes, how long to spend transmitting the SMTP command RCPT TO:. The default is 10.

STATUS_DATA_RECEIVE_TIME (integer)
This option specifies, in minutes, how long to wait to receive the SMTP response to our sent data; *i.e.*, how long to wait to receive a "250" (or other) response to the dot terminating sent data. The default value is 10.

See also the STATUS_DATA_RECV_PER_ADDR_TIME, STATUS_DATA_RECV_PER_BLOCK_TIME, and STATUS_DATA_RECV_PER_ADDR_PER_BLOCK_TIME options.

STATUS_DATA_RECV_PER_ADDR_TIME (floating point value)
This option specifies an adjustment factor for how long to wait to receive the SMTP response to our sent data based on the number of addresses in the MAIL TO: command. This value is multiplied by the number of addresses and added to the base wait time (specified with the STATUS_DATA_RECEIVE_TIME option). The default is *0.083333*.

STATUS_DATA_RECV_PER_BLOCK_TIME (floating point value)
This option specifies an adjustment factor for how long to wait to receive the SMTP response to our sent data based on the number of blocks sent. This value is multiplied by the number of blocks and added to the base wait time (specified with the STATUS_DATA_RECEIVE_TIME option). The default is 0.001666.

STATUS_DATA_RECV_PER_ADDR_PER_BLOCK_TIME (floating point value)
This option specifies an adjustment factor for how long to wait to receive the SMTP response to our sent data based on the number of addresses (in the MAIL TO: command) per number of blocks sent. This value is multiplied by the number of addresses per block and added to the base wait time (specified with the STATUS_DATA_RECEIVE_TIME option). The default is 0.003333.

STATUS_MAIL_RECEIVE_TIME (integer)
This option specifies, in minutes, how long to wait to receive the SMTP response to a sent MAIL FROM: command. The default is 10.

STATUS_RCPT_RECEIVE_TIME (integer)
This option specifies, in minutes, how long to wait to receive the SMTP response to a sent RCPT TO: command. The default value is 10.

STATUS_RECEIVE_TIME (integer)

This option specifies, in minutes, how long to wait to receive the SMTP reply to general SMTP commands, (replies other than those with explicitly specified time out values set using other specifically named options). The default value is 10.

STATUS_TRANSMIT_TIME (integer)

This option specifies, in minutes, how long to spend transmitting the SMTP reply to an SMTP command. The default value is 10.

TRACE_LEVEL (0 or 2)

This option controls whether TCP/IP level trace is included in debug log files. The default value is 0, meaning that no TCP/IP packet traces are included; a value of 2 tells PMDF to include TCP/IP packet traces in any debug log files and to include some additional information, such as DNS lookup information, in addition to the basic TCP/IP packet traces.

21.1.3 Replacing the Native SMTP Server with PMDF's SMTP Server

On UNIX, the existing SMTP server should be replaced by PMDF's SMTP server during the installation of PMDF; see the appropriate edition of the *PMDF Installation Guide*.

On NT, any existing SMTP server must be disabled, so that the PMDF SMTP server can replace it. For instance, Windows 2000 Server and Advanced Server do have a native SMTP server by default. See the appropriate edition of the *PMDF Installation Guide* for more information.

On OpenVMS, your TCP/IP package's native SMTP server must be replaced by PMDF's SMTP server. First the native SMTP server must be disabled, as described below, then the PMDF Service Dispatcher must be configured to handle the SMTP service, as described in Section 21.1.4, and then the Service Dispatcher must be restarted, or started if it was not already running, to start up the new service.

Disabling the HP TCP/IP SMTP server, or a PMDF UTCP SMTP server

To disable the SMTP server native to HP TCP/IP Services, (also known as UCX), or a previous PMDF UTCP channel, issue the commands:

```
$ UCX
UCX> DISABLE SERVICE SMTP
UCX> SET NOSERVICE SMTP
UCX> SET CONFIGURATION ENABLE NOSERVICE SMTP
UCX> EXIT
```

If your site was previously using the PMDF `UCX_SMTPD` dispatcher, you must also delete that process; it will have a process name of "UCX/PMDF server". Also remove any command from your system startup that starts up that process; *e.g.*, remove any commands such as

```
$ @PMDF_COM:start_ucx_smtpd.com
```

Disabling the TCPware SMTP server, or a PMDF PTCP SMTP server

To disable the SMTP server native to Process Software TCPware TCP/IP, or a previous PMDF PTCP channel, issue the commands:

```
$ RUN TCPWARE:netcu
NETCU REMOVE SERVICE 25 TCP
NETCU EXIT
```

Or if you are running Process Software TCPware TCP/IP V5.3 (or higher), you can disable the SMTP server using the interactive configuration tool,

```
$ @TCPWARE:CNFNET TCP
```

by answering NO to the question regarding using the SMTP server, along the lines of:

```
Do you want to use the SMTP Mail Transfer Agent?
```

Note that if you have also installed Process Software's SMTP-VMS optional product, you must disable it by issuing the following command:

```
$ RENAME TCPWARE:smtp_control.com -
_$       TCPWARE:smtp_control.com_disabled
```

Once you have disabled the SMTP server native to Process Software TCPware TCP/IP, or a previous PMDF PTCP channel, TCPware should then be shutdown and restarted with the commands:

```
$ @TCPWARE:shutnet.com
$ @TCPWARE:startnet.com
```

If you have TCPware installed in a cluster environment you must restart TCPware on every node in the cluster to ensure that each node knows about the change.

Disabling the Multinet SMTP server, or a PMDF MTCP SMTP server

To disable the SMTP server native to Process Software's MultiNet TCP/IP, issue the following commands:

```
$ MULTINET CONFIGURE/SERVERS
SERVER-CONFIG> DISABLE SMTP
SERVER-CONFIG> RESTART
SERVER-CONFIG> EXIT
```

If you have MultiNet installed in a cluster environment you must restart MultiNet on every node in the cluster to ensure that each node knows about the change.

21.1.4 Configuring the PMDF Service Dispatcher to Handle the SMTP Service

The multithreaded TCP SMTP channel's SMTP server is designed to be handled by the PMDF Service Dispatcher. The Service Dispatcher creates and uses worker processes that handle the SMTP service; see Chapter 11 for more details. In order to use the multithreaded TCP SMTP channel, the Service Dispatcher must be configured to handle the SMTP service, if it has not been so configured already. This configuration can be achieved using the PMDF GUI configuration utility as described in the appropriate edition of the *PMDF Installation Guide*, or by using the command line utility `pmdf configure dispatcher` (OpenVMS and UNIX); if you have not already configured the PMDF Dispatcher you should do so at this time.

21.2 Controlling the SMTP Server

The multithreaded TCP SMTP channels' SMTP server is handled by the PMDF Service Dispatcher. Each worker process the Service Dispatcher creates for the multithreaded SMTP server can handle multiple, simultaneous connections, and the Service Dispatcher can create multiple such worker processes. To start the SMTP server, you must start the PMDF Service Dispatcher.

Note: Be certain to disable any other SMTP server, *e.g.*, a native SMTP server shipped with a TCP/IP package, as described above in Section 21.1.3 *before* asking the Service Dispatcher to start the multithreaded SMTP server; the Service Dispatcher will not be able to start the multithreaded SMTP server if another process has already bound to port 25.

The Service Dispatcher can be started with the command

```
pmdf startup dispatcher
```

If you modify your PMDF configuration or options that apply to the multithreaded TCP SMTP server, you must restart the server so that the new configuration or options will take effect. On OpenVMS and UNIX platforms, use the command

```
pmdf restart smtp
```

On NT platforms, you must restart the Dispatcher itself using the command

```
C:\> pmdf restart dispatcher
```

A new SMTP server process will be created, using the new configuration, and will process subsequent incoming SMTP connections. The old TCP SMTP server process will finish up any SMTP sessions it might have and exit when they are finished.

Note that you can stop the TCP SMTP server at any time. On OpenVMS and UNIX platforms, use the command

```
pmdf shutdown smtp
```

which will shut down the server gracefully, allowing any outstanding connections to finish up.

On NT, you must shut down the Dispatcher itself, or edit the Dispatcher configuration removing the SMTP service and then restarting the Dispatcher.

21.2.1 SMTP Connection Control Mapping

The PMDF Service Dispatcher is able to selectively accept or reject incoming SMTP connections based on IP address and port number. For instance, incoming connections to the SMTP port(s) (normally port 25) can be blocked based on the sending system's IP number; see Section 11.5 for details.

21.3 Accessing Gateway Systems

A local TCP/IP network can include one or more systems that are equipped to relay messages to machines not directly accessible on the local network. Such gateway systems accept addresses that are not palatable to the network itself.

One solution to this problem is to use appropriate MX records and a name resolver. However, this approach can be infeasible in some environments, so a different solution can be needed.

There is an alternate approach, in which routing to TCP/IP gateways is done by creating additional channels, one per gateway system, in the configuration file. The name of these channels must always begin with `tcp_` for the multithreaded TCP SMTP channel. The channel blocks have the general form:

```
tcp_gateway smtp daemon router
gateway-system-name
```

or equivalently,

```
tcp_gateway smtp daemon gateway-system-name
arbitrary-placeholder-name
```

Rewrite rules must then be added to the configuration file to route the appropriate addresses to the gateway. See, for instance, Example 2–3.

The `daemon router` keyword phrase tells the SMTP client program not to open a connection directly to the first system named in the envelope address list, but to instead open a connection to the official host for this channel, *gateway-system-name*. Certain gateways can restrict the number of addresses that can appear in a single copy of a message so it can be appropriate to add the `single` or `single_sys` keywords to the channel block for some gateway channels. If the gateway can handle multiple simultaneous connections, then use of the `threaddepth` keyword can be of interest to cause outgoing connections to be split amongst multiple threads.

Once a channel block for a gateway is created the channel should be ready to use.

21.4 Triggering (on-demand) Message Transfer with Remote Systems

In cases where the network connection between two systems is only available at particular times—a "dial up" sort of connection for instance—there is an SMTP extension whereby one system can inform another that it is ready to receive mail. This is performed using the SMTP extension command ETRN, defined in RFC 1985: [6] the side that desires to receive mail connects to the remote side's SMTP server and issues the command ETRN *receivinghostname*. If the remote side's SMTP server supports the ETRN command, it will then attempt delivery of any messages it has waiting to be sent to *receivinghostname*.

The PMDF SMTP server supports ETRN. In particular, the PMDF SMTP server interprets a received ETRN *domainname* command as a request to deliver all messages to *domainname*, a received ETRN @*domainname* as a request to deliver all messages in the *domainname* subnet, and a ETRN #*channelname* command as a request to run the channel *channelname*. By default, the PMDF SMTP always responds to a remote site's ETRN requests; if you want to restrict this behavior, see Section 2.3.4.34.

And outgoing PMDF SMTP-based channels, such as TCP/IP channels, can be configured to send an ETRN command at the beginning of an outgoing SMTP dialogue via the sendetrn channel keyword; see Section 2.3.4.33. For instance, suppose a system host1.example.com has a dial-up connection to a remote system intermittent.some.where.com, where the intermittent.some.where.com system also supports ETRN. For a channel for connecting up to the remote side and sending ETRN, such a site might use a channel definition along the lines of:

```
tcp_dialup smtp mx daemon intermittent.some.where.com \
 periodic sendetrn host1.example.com
TCP-DIALUP
```

21.5 SASL Authentication for the TCP/IP Channel Client

PMDF has the ability to configure the TCP/IP channel client to use SASL via the SMTP AUTH command when sending mail out from the PMDF MTA to a remote MTA. This is primarily needed by home users who are running PMDF on their home systems and have an ISP that requires a username and password to be able to send out mail through the ISP's MTA.

The username and password to use for authentication is configured in a section in the security.cnf file called CLIENT_AUTH. For details see Section 14.2. An example default CLIENT_AUTH section is as follows:

```
[CLIENT_AUTH=default]
USER=remote-username
PASSWORD=remote-password
```

[6] If installing the RFCs was chosen as an option during the PMDF installation, you will have a copy of this RFC on your system in the directory pmdf_root:[doc.rfc] (OpenVMS) or /pmdf/doc/rfc/ (UNIX).

The TCP/IP channel also needs to be configured to enable client-side SASL. This is done with one of the following channel keywords: `maysaslclient`, `mustsaslclient`, `maysasl`, or `mustsasl`. For details see Section 2.3.4.43.

By default, the `[CLIENT_AUTH=default]` section in the `security.cnf` file is used to get the username and password. To use a different `CLIENT_AUTH` section, specify its name using the `client_auth` channel keyword.

This example channel definition is used to send mail out to a system called 'alpha' on the SMTP submission port (587) using SASL and TLS.

```
tcp_alpha smtp mx port 587 daemon router maysaslclient allowswitchchannel \
        maytls client_auth alpha
alpha.example.edu
TCP-ALPHA
```

22 Message Manipulation Channels

This chapter describes how to use the conversion, script, and disclaimer channels to modify messages as they pass through PMDF.

The conversion channel allows processing of message attachments and other body parts using external, third-party programs and site-supplied procedures, such as virus scanners. The execution of such procedures is controlled by the CONVERSIONS mapping table and the conversions file. The conversion channel is discussed in Section 22.1.

The PMDF conversions file, discussed in Section 22.1.3, is used to specify the details of conversion channel processing and to specify the details of some internal CHARSET-CONVERSION table triggered conversions (see Chapter 6).

Note: If you use both a CONVERSIONS mapping table to invoke the conversion channel and a CHARSET-CONVERSION mapping table to perform character set conversions (see Chapter 6), the CONVERSIONS mapping table takes precedence. In this case, all entries with RELABEL=1 in the conversions file are skipped. A conversions file entry can be changed from being associated with the CHARSET-CONVERSION mapping table to being associated with the CONVERSIONS mapping table by replacing the RELABEL=1 parameter with the COMMAND parameter. The command specified could be as simple as copying the input file to the output file.

The script channel allows processing of messages in their entirety using external, third-party programs and site-supplied procedures. The execution of such procedures is controlled by the SCRIPT mapping table. The script channel is discussed in Section 22.2.

The disclaimer channel allows arbitrary text (usually in the form of a disclaimer) to be added to messages. The addition of such text is controlled by the DISCLAIMER mapping table. The disclaimer channel is discussed in Section 22.3.

22.1 Conversion Channel

The conversion channel performs arbitrary body part by body part processing on messages flowing through PMDF. Any subset of PMDF traffic can be selected to pass through the conversion channel and any set of programs or command procedures can be used to perform conversion channel processing.

For instance, third party document convertors or virus scanning software may be hooked in for automatic execution via the conversion channel. Or sites may develop their own custom applications to hook in.

22.1.1 Selecting Traffic for Conversion Processing

Although conversion channel processing is done using a regular PMDF channel program, under normal circumstances this channel is never specified directly either in an address or in a PMDF rewrite rule. PMDF controls access to the `conversion` channel via the `CONVERSIONS` mapping table in the PMDF mappings file.

As PMDF processes each message it probes the `CONVERSIONS` mapping (if one is present) with a string of the form

`IN-CHAN=`*`source-channel`*`;OUT-CHAN=`*`destination-channel`*`;CONVERT`

where *`source-channel`* is the source channel from which the message is coming and *`destination-channel`* is the destination channel to which the message is heading. If a match occurs the resulting string should be a comma-separated list of keywords. Table 22–1 lists the available keywords.

Note: Make sure that there is no whitespace in the resulting string, for example around commas or equal signs.

Table 22–1 CONVERSIONS Mapping Table Keywords

Keyword	Action
Channel=*channel*	Enables conversion channel processing using the conversion channel named *channel*. Note that technically, the channel specified can be any channel defined in `pmdf.cnf`.
No	Disables conversion channel processing.
Yes	Enables conversion channel processing.

A `No` is assumed if no match occurs.

If the `CONVERSIONS` mapping table enables the conversion channel, PMDF diverts the message from its regular destination to the conversion channel. If the conversion channel is not enabled, the message is queued to its regular destination channel.

For example, suppose messages require command channel processing if they come from outside your organization and are destined for either local users or remote MAIL-11 (DECnet) users. The following mapping would then be appropriate:

```
CONVERSIONS

  IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT          Yes
  IN-CHAN=tcp_local;OUT-CHAN=d;CONVERT          Yes
  IN-CHAN=*;OUT-CHAN=*;CONVERT                  No
```

Note: The CONVERSIONS mapping table is not checked for messages which have already been discarded, for example by a mailbox filter.

22.1.2 Configuration

The first step is to add the conversion channel to the PMDF configuration file. The entry should have the form:

```
conversion
CONVERSION-DAEMON
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the conversion channel. Something like

```
conversion                      $U%conversion.localhostname@CONVERSION-DAEMON
conversion.localhostname        $U%conversion.localhostname@CONVERSION-DAEMON
```

where *localhostname* is the name of the local PMDF system, provides the necessary functionality. Once this is done, addresses of the form

user%host@conversion.*localhostname*

are routed through the conversion channel regardless of what the CONVERSIONS mapping says.

Additional conversion_* channels may be defined. For example:

```
conversion_2
CONVERSION2-DAEMON
```

These alternate conversion channels are generally selected using the CONVERSIONS mapping table Channel keyword (see Table 22–1). If desired, you may also create separate rewrite rules for each of these alternate conversion channels, similar to the ones shown above.

Note: A channel named conversion **must** be defined to enable conversion channel processing. This channel is used as the default conversion channel.

Note that the conversion channel and all conversion_* channels use a single conversions file (see Section 22.1.3).

22.1.3 Conversion Control

The actual processing performed by the conversion channel is controlled by rules specified in the PMDF conversions file. The conversions file is located via the PMDF_CONVERSION_FILE logical name (OpenVMS), or PMDF tailor file option (UNIX), or Registry entry (NT), and is usually the file pmdf_table:conversions. on OpenVMS, or /pmdf/table/conversions on UNIX, or C:\pmdf\table\conversions on NT.

The PMDF conversion file is a text file containing entries in a format that is modelled after MIME Content-Type: parameters. Each entry consists of one or more lines grouped together; each line contains one or more name=value; parameter clauses. Quoting rules conform to MIME conventions for Content-Type: header line parameters. Every line except the last must end with a semicolon. Entries are

terminated by either a line that does not end in a semicolon, one or more blank lines, or both.

For example, the following entry (on OpenVMS) specifies that all message parts should be run through a site-supplied virus checking command procedure:

```
in-channel=*; in-type=*; in-subtype=*;
  parameter-symbol-0=NAME; parameter-copy-0=*;
  dparameter-symbol-0=FILENAME; dparameter-copy-0=*;
  message-header-file=2; original-header-file=1;
  override-header-file=1; override-option-file=1;
  command="@pmdf_com:check-virus.com/output=pmdf_log:check-virus.out"
```

22.1.3.1 Conversion Entry Scanning and Application

The conversion channel processes each message part by part. The header of each part is read and its Content-Type: and other header information is extracted. The entries in the conversion file are then scanned in order from first to last; any IN- parameters present and the OUT-CHAN parameter, if present, are checked. If all of these parameters match the corresponding information for the body part being processed, then the conversion specified by the remainder of the parameters is performed. Note that an entry must include an IN-TYPE clause in order to match. More specifically, the matching checks:

- if the IN-CHAN and OUT-CHAN parameters match the channels through which the message is passing; and
- if the PART-NUMBER matches the structured part number[7] of the message part; and
- if all of the IN-PARAMETER-NAME, IN-PARAMETER-VALUE, IN-SUBTYPE, and IN-TYPE, parameters match the Content-Type of the message; and
- if all of the IN-DISPOSITION, IN-DPARAMETER-NAME, and IN-DPARAMETER-VALUE parameters match the Content-Disposition of the message; and
- if the IN-DESCRIPTION matches the Content-Description of the message; and
- if the IN-SUBJECT, IN-A1-TYPE, and IN-A1-FORMAT match the headers of the immediately enclosing message message/rfc822 part.

Only if all specified parameters match is the entry considered to match. Scanning terminates once a matching entry has been found or all entries have been exhausted. If no entry matches, no conversion is performed.

If the matching entry specifies DELETE=1, then the message part is deleted. Otherwise, the command specified by the COMMAND parameter is executed.

Once an entry with a COMMAND parameter has been selected the body part is extracted to a file (on OpenVMS, in a manner specified by the FDL-OVERRIDE and FDL-STRING parameters). The converter execution environment is prepared as specified by the PARAMETER-SYMBOL-*n* and DPARAMETER-SYMBOL-*n* parameters, as well as all of the OUT- parameters. The OUT-TYPE, OUT-SUBTYPE, OUT-DESCRIPTION, OUT-DISPOSITION, and OUT-ENCODING parameters are used to specify the initial values for

[7] The structured part number is the message part number as it would appear in PMDF MAIL.

the corresponding DCL symbols (OpenVMS) or environment variables (UNIX and NT). The OUT-PARAMETER-NAME-*n*, OUT-PARAMETER-VALUE-*n*, OUT-DPARAMETER-NAME-*n*, and OUT-DPARAMETER-VALUE-*n* parameters are used to modify the Content-Type: and Content-Disposition: headers, which are passed to the command in the INPUT_HEADERS file.

Finally, a subprocess is created to run the command specified by the COMMAND parameter. The command should perform the necessary conversion operation, reading the file specified by the INPUT_FILE DCL symbol (OpenVMS) or environment variable (UNIX and NT) and producing the file specified by the OUTPUT_FILE DCL symbol (OpenVMS) or environment variable (UNIX and NT). The command may optionally specify its own MIME headers in the file specified by the OUTPUT_HEADERS DCL symbol (OpenVMS) or environment variable (UNIX and NT).

Note: The file specified by OUTPUT_FILE **must** be created by the command procedure (OpenVMS) or script (Unix and Windows). If you do not wish to make any changes, you must copy INPUT_FILE to OUTPUT_FILE. If you do not supply an OUTPUT_FILE then the part will be deleted.

On OpenVMS, the command may optionally define job table logical names to pass information back to the conversion channel.

On UNIX and NT, the command may optionally set options in the OUTPUT_OPTIONS file to pass information back to the conversion channel.

Conversion operations are terminated and no conversion is performed if the spawned command returns an error.

If the command succeeds, the output symbols (OpenVMS) or options (UNIX and NT) are processed, the resulting output file is read as specified by OUTPUT_MODE, and if the OVERRIDE-HEADER-FILE parameter was set to 1, the output header file is read. A new body part containing the converted material and converted header is constructed according to the specified output symbols OUTPUT_TYPE, OUTPUT_SUBTYPE, OUTPUT_DESCRIPTION, OUTPUT_DISPOSITION, OUTPUT_ENCODING.

This process is repeated for each part of the message until all parts have been processed.

22.1.3.2 Available Parameters

The rule parameters currently provided are shown in Table 22–2.

Table 22–2 Available Conversion Parameters

	Parameter	Meaning
	COMMAND	Command to execute. This parameter is required; if no command is specified, the entry is ignored. Note: On NT, if the command contains a backslash, it must be escaped with another backslash. For example: C:\\PMDF\\TABLE\\TEST.BAT. [1]
	DELETE	0 or 1. If this flag is set, the message part is deleted. (If this is the only part in a message, then a single empty text part is substituted.)
	DPARAMETER-COPY-*n*	A list of the `Content-Disposition:` parameters to copy from the input body part's `Content-Disposition:` parameter list to the output body part's `Content-Disposition:` parameter list; *n* = 0, 1, 2, ... Takes as argument the name of the MIME parameter to copy, as matched by an `IN-PARAMETER-NAME-m` clause. Wildcards may be used in the argument. In particular, an argument of `*` means to copy all the original `Content-Disposition:` parameters.
	DPARAMETER-SYMBOL-*n*	`Content-disposition` parameters to convert to environment variables (DCL symbols on OpenVMS) if present; *n* = 0, 1, 2, ... Takes as argument the name of the MIME parameter to convert, as matched by an `IN-DPARAMETER-NAME-m` clause. Each `DPARAMETER-SYMBOL-n` is extracted from the `Content-Disposition:` parameter list and placed in an environment variable or DCL symbol of the same name prior to executing the command.
†	FDL-OVERRIDE	0 or 1; if 1, the FDL information specified by the FDL-STRING entry parameter is used unconditionally; if 0, the `FDL-STRING` entry parameter is overriden by an `FDL-STRING Content-Type:` parameter.
†	FDL-STRING	OpenVMS FDL information used to construct the input file for the command. Prior to executing the command, the body part is written into an RMS file created using this FDL information. An `FDL-STRING` parameter overrides the `FDL-STRING` entry parameter unless `FDL-OVERRIDE` is 1. Usage of this parameter is restricted to OpenVMS systems.
	IN-A1-FORMAT	Input A1-Format from enclosing `message/rfc822` part.
	IN-A1-TYPE	Input A1-Type from enclosing `message/rfc822` part.
	IN-CHAN	Input channel to match (wildcards allowed). The actions specified by this entry are only performed if the message is coming from the specified channel.
	IN-CHANNEL	Synonym for `IN-CHAN`.
	IN-DESCRIPTION	Input MIME `Content-Description` header.
	IN-DISPOSITION	Input MIME `Content-Disposition` header.
	IN-DPARAMETER-DEFAULT-*n*	Input MIME `Content-Disposition` parameter value default if parameter is not present. This value is used as a default for the `IN-DPARAMETER-VALUE-n` test when no such parameter is specified in the body part.
	IN-DPARAMETER-NAME-*n*	Input MIME `Content-Disposition` parameter name whose value is to be checked; *n* = 0, 1, 2, ...
	IN-DPARAMETER-VALUE-*n*	Input MIME `Content-Disposition` parameter value that must match corresponding `IN-DPARAMETER-NAME` (wildcards allowed). The actions specified by this entry are only performed if this field matches the corresponding parameter in the body part's `Content-Disposition:` parameter list.

[1]Except see the RELABEL and SERVICE-COMMAND parameters, which cause entries to be ignored during conversion channel processing, but do affect character set conversion.

†Supported only on OpenVMS; ignored on UNIX and NT.

Table 22–2 (Cont.) Available Conversion Parameters

Parameter	Meaning
IN-PARAMETER-DEFAULT-*n*	Input MIME `Content-Type` parameter value default if parameter is not present. This value is used as a default for the `IN-PARAMETER-VALUE-n` test when no such parameter is specified in the body part.
IN-PARAMETER-NAME-*n*	Input MIME `Content-Type` parameter name whose value is to be checked; *n* = 0, 1, 2, ...
IN-PARAMETER-VALUE-*n*	Input MIME `Content-Type` parameter value that must match corresponding `IN-PARAMETER-NAME` (wildcards allowed). The actions specified by this entry are only performed if this field matches the corresponding parameter in the body part's `Content-Type:` parameter list.
IN-SUBJECT	Input Subject from enclosing `message/rfc822` part.
IN-SUBTYPE	Input MIME subtype to match for conversion (wildcards allowed). The actions specified by this entry are only performed if this field matches the MIME subtype of the body part.
IN-TYPE	Input MIME type to match (wildcards allowed). The actions specified by this entry are only performed if this field matches the MIME type of the body part.
MESSAGE-HEADER-FILE	0, 1, or 2. If set to 1, the original headers of the immediately enclosing message part are written to the file represented by the `MESSAGE_HEADERS` symbol. If set to 2, the original headers of the message as a whole (the outermost message headers) are written to `MESSAGE_HEADERS`.
ORIGINAL-HEADER-FILE	0 or 1. If set to 1, the original headers of the enclosing part are written to the file represented by the `INPUT_HEADERS` symbol.
OUT-A1-FORMAT	Output A1-Format.
OUT-A1-TYPE	Output A1-Type.
OUT-CHAN	Output channel to match (wildcards allowed). The actions specified by this entry are only performed if the message is destined for the specified channel.
OUT-CHANNEL	Synonym for `OUT-CHAN`.
OUT-DESCRIPTION	Output MIME `Content-Description` if it is different than the input MIME `Content-Description`.
OUT-DISPOSITION	Output MIME `Content-Disposition` if it is different than the input MIME `Content-Disposition`
OUT-DPARAMETER-NAME-*n*	Output MIME `Content-Disposition` parameter name; *n* = 0, 1, 2, ...
OUT-DPARAMETER-VALUE-*n*	Output MIME `Content-Disposition` parameter value corresponding to `OUT-DPARAMETER-NAME-n`.
OUT-MODE	Mode in which to read the output file. This should be one of: `BLOCK`, `RECORD`, `RECORD-ATTRIBUTE`, or `TEXT`.
OUT-ENCODING	Encoding to apply to the output file.
OUT-PARAMETER-NAME-*n*	Output MIME `Content-Type` parameter name; *n* = 0, 1, 2, ...
OUT-PARAMETER-VALUE-*n*	Output MIME `Content-Type` parameter value corresponding to *OUT-PARAMETER-NAME-n*.
OUT-SUBTYPE	Output MIME type if it is different than the input MIME type.
OUT-TYPE	Output MIME type if it is different than the input type.

Table 22–2 (Cont.) Available Conversion Parameters

	Parameter	Meaning
	OVERRIDE-HEADER-FILE	0 or 1. If set, then MIME headers are read from the OUTPUT_HEADERS symbol, overriding the original MIME headers in the enclosing part.
‡	OVERRIDE-OPTION-FILE	0 or 1. If set, then the conversion channel reads options from the OUTPUT_OPTIONS symbol.
	PARAMETER-COPY-*n*	A list of the `Content-Type:` parameters to copy from the input body part's `Content-Type:` parameter list to the output body part's `Content-Type:` parameter list; *n* = 0, 1, 2, ... Takes as argument the name of the MIME parameter to copy, as matched by an `IN-PARAMETER-NAME-m` clause. Wildcards may be used in the argument. In particular, an argument of `*` means to copy all the original `Content-Type:` parameters.
	PARAMETER-SYMBOL-*n*	Content-Type parameters to convert to environment variables (DCL symbols on OpenVMS) if present; *n* = 0, 1, 2, ... Takes as argument the name of the MIME parameter to convert, as matched by an `IN-PARAMETER-NAME-m` clause. Each `PARAMETER-SYMBOL-n` is extracted from the `Content-Type:` parameter list and placed in an environment variable or DCL symbol of the same name prior to executing the command.
	PART-NUMBER	Dotted integers, *e.g.*, *a.b.c*... The part number of the MIME body part.
	RELABEL	0 or 1. This flag causes an entry to be ignored during conversion channel processing. However, if this flag is 1, then MIME header relabelling is performed during character set conversion.[2]
	SERVICE-COMMAND	The command to execute to perform service conversion. This flag causes an entry to be ignored during conversion channel processing. `SERVICE-COMMAND` entries are instead performed during character set conversion processing.[3]
	TAG	Input tag, as set by a mailing list `[CONVERSION_TAG]` named parameter, must match.

[2]See Section 6.3.2 for more information on character set conversion and using the RELABEL parameter.

[3]See Section 6.4 for more information on character set conversion and using the SERVICE-COMMAND parameter.

‡Available on UNIX and NT only

Parameters not listed in the preceeding table are ignored.

22.1.3.3 Conversion Entry Parameter Value Wildcard Matching

The values of conversion entry parameter values may be specified as literal strings, or using wildcards as in PMDF mapping entry patterns. See Section 5.3.1 for a discussion of available wildcards.

For instance,

```
in-dparameter-name-0=filename; in-dparameter-value-0=*.wpc;
```

would match any `Content-disposition:` header filename parameter that has a ".wpc" extension. Or

```
in-dparameter-name-0=filename; in-dparameter-value-0=*.wp$[cd56]%;
```

would match any `Content-disposition:` header filename parameter that has a ".wpc", ".wpd", ".wp5", or ".wp6" extension.

22.1.3.4 Predefined Symbols or Environment Variables

Table 22–3 shows the basic set of DCL symbols (OpenVMS) or environment variables (UNIX and NT) available for use by the `conversion` command.

Table 22–3 Symbols for Use by the Conversion Channel

	Symbol	Description
	INPUT_ENCODING	The encoding originally present on the body part.
	INPUT_FILE	The name of the file containing the original body part. The command should read this file.
	INPUT_HEADERS	The name of the file containing the original headers for the enclosing part. The command should read this file.
	INPUT_TYPE	The content type of the input message part.
	INPUT_SUBTYPE	The content subtype of the input message part.
	INPUT_DESCRIPTION	The content description of the input message part.
	INPUT_DISPOSITION	The content disposition of the input message part.
	MESSAGE_HEADERS	The name of the file containing the original headers for an enclosing (outermost) message. The command should read this file.
	OUTPUT_FILE	The name of the file where the command should store its output. The command should create and write this file.
	OUTPUT_HEADERS	The name of the file where the command should store MIME headers for an enclosing part. The command should create and write this file. Note that the file should have a format of header line, header line,..., blank line; be sure to include the final blank line.
†	OUTPUT_OPTIONS	The name of the file to which the command should write options (such as status values).

†Available on UNIX and NT only

Additional symbols containing `Content-Type:` information can be created as they are needed using the `PARAMETER-SYMBOL-n` facility.

Table 22–4 shows additional symbols (OpenVMS) or "override" options (UNIX and NT) available for use by the conversion channel. The command procedure or script may use these to pass information back to the conversion channel. To set these symbols on OpenVMS, the command procedure should define corresponding logical names using the DCL command DEFINE/JOB. To set these options on UNIX or NT, specify `OVERRIDE-OPTION-FILE=1` in the desired conversion entry and then have the script write the desired options to the `OUTPUT_OPTIONS` file.

Table 22–4 Symbols (OpenVMS) or Options (UNIX and NT) for Passing Information Back to the Conversion Channel

	Symbol or option	Description
	OUTPUT_TYPE	The content type of the output message part.
	OUTPUT_SUBTYPE	The content subtype of the output message part.
	OUTPUT_DESCRIPTION	The content description of the output message part.
	OUTPUT_DIAGNOSTIC	Text to include in the error text returned to the message sender if a message is forcibly bounced (via `PMDF__FORCERETURN`) by the conversion channel.
	OUTPUT_DISPOSITION	The content disposition of the output message part.
	OUTPUT_ENCODING	The content transfer encoding to use on the output message part.
	OUTPUT_MODE	The mode with which the conversion channel should write the output message part, hence the mode with which recipients should read the output message part.
†	STATUS	The PMDF completion status for the script.

†Available on UNIX and NT only

22.1.3.5 Symbol Substitution in Conversion Entries

Certain values from the body part being processed may be substituted into a conversion entry by enclosing a corresponding symbol name in single quotes. Table 22–5 shows the list of symbols that can be used.

Table 22–5 Conversion Symbols for Substitution

Symbol	Description
A1-FORMAT	The value of the `A1-Format:` header.
A1-FUNCTION	The value of the `A1-Function:` header.
A1-TYPE	The value of the `A1-Type:` header.
DESCRIPTION	The value of the `Content-Description:` header.
DISPOSITION	The primary value of the `Content-Disposition:` header (for example `attachment`).
LANGUAGE	The value of the `Content-Language:` header.
SUBJECT	The subject of the message.
SUBTYPE	The content subtype of the message part.
TAG	The value of the input tag, as set by a mailing list [CONVERSION_TAG] named parameter.
TYPE	The content type of the message part.

In addition to the symbols listed in Table 22–5, any parameter from the `Content-Type:` header (for example, `name`), or the `Content-Disposition:` header (for example, `filename`) may be specified.

Note: For the string value of the COMMAND parameter only, any of the symbols listed in Table 22–3 may be substituted using the standard command line symbol substitution for the given platform, *i.e.*, preceding and following the variable's name with an apostrophe on OpenVMS, preceding the variable's name with a dollar character on UNIX, or preceding and following the variable's name with a percent sign on NT.

For example, with a site-supplied command procedure pmdf_table:site.com that attempts to perform various operations, one might use an entry along the lines of:

```
in-chan=tcp_local; out-chan=l; in-type=application; in-subtype=*;
  out-type='TYPE'; out-subtype='SUBTYPE';
  command="@pmdf_table:site.com 'INPUT_FILE' 'OUTPUT_FILE' 'INPUT_TYPE' 'INPUT_SUBTYPE'"
```

To obtain a literal single quote in a conversion entry, quote it with the backslash character, \'. To obtain a literal backslash in a conversion entry, use two backslashes, \\.

22.1.3.6 Calling Out to a Mapping Table from a Conversion Entry

The value for a conversion parameter may be obtained by calling out to a mapping table. The syntax for calling out to a mapping table is

```
"'mapping-table-name:mapping-input'"
```

For instance, with a mapping table

```
X-ATT-NAMES

   postscript            PS.PS$Y
   wordperfect5.1        WPC.WPC$Y
   msword                DOC.DOC$Y
```

then on OpenVMS a conversion entry such as the following results in substituting generic file names in place of specific file names on attachments.

```
out-chan=tcp_local; in-type=application; in-subtype=*;
  in-parameter-name-0=name; in-parameter-value-0=*:[*]*;
  out-type=application; out-subtype='SUBTYPE';
  out-parameter-name-0=name;
  out-parameter-value-0="'X-ATT-NAMES:\\'SUBTYPE\\''";
  command="COPY 'INPUT_FILE' 'OUTPUT_FILE'"
```

Or on UNIX, a conversion entry such as the following results in substituting generic file names in place of specific file names on attachments.

```
out-chan=tcp_local; in-type=application; in-subtype=*;
  in-parameter-name-0=name; in-parameter-value-0=/*/*;
  out-type=application; out-subtype='SUBTYPE';
  out-parameter-name-0=name;
  out-parameter-value-0="'X-ATT-NAMES:\\'SUBTYPE\\''";
  command="cp $INPUT_FILE $OUTPUT_FILE"
```

22.1.3.7 The Headers in an Enclosing Part or Message

When performing conversions on a message part, the conversion channel has access to the headers in an enclosing part, an enclosing message/rfc822 part, or to the outermost message headers if there is no enclosing message/rfc822 part.

For instance, the IN-A1-TYPE and IN-A1-FORMAT parameters can be used to check the A1-Type and A1-Format headers of an enclosing part, and the OUT-A1-TYPE and OUT-A1-FORMAT parameters can be used to set those enclosing headers. Or decisions about interior message part processing can be made based upon the message's outermost headers.

More generally, if an entry is selected that has ORIGINAL-HEADER-FILE=1, then the headers of that part are written to the file represented by the INPUT_HEADERS symbol. If an entry is selected that has MESSAGE-HEADER-FILE=1, then all the original headers of the enclosing message/rfc822 part are written to the file represented by the MESSAGE_HEADERS symbol. Or if an entry is selected that has MESSAGE-HEADER-FILE=2, then all the original headers of the outermost message are written to the file represented by the MESSAGE_HEADERS symbol.

Note that the envelope From: information is included in the MESSAGE_HEADERS file in the X-Envelope-From: header, and the envelope To: information is in the X-Envelope-To: header. However, if you have specified the nox_env_to keyword on your conversion channel definition, these headers will not be included.

If OVERRIDE-HEADER-FILE=1, then the conversion channel reads and uses as the headers on that enclosing part the contents of the file represented by the OUTPUT_HEADERS symbol.

22.1.4 Command Completion Statuses

The command procedure or script specified in the conversions file entry should return one of the following completion statuses. On OpenVMS, this is accomplished by exiting from the command procedure with the desired status. On UNIX and Windows, this is accomplished by setting the STATUS option in the OUTPUT_OPTIONS file. The script itself should exit explicitly with a value of 0, otherwise the conversion channel will think that the running of the script failed. Note that you must specify OVERRIDE-OPTION-FILE=1 in the conversion entry to enable the OUTPUT_OPTIONS file.

Table 22–6 shows the list of completion statuses that can be specified. The values for these statuses are defined in pmdf_com:pmdf_err.h (OpenVMS), /pmdf/include/pmdf_err.h (UNIX), or C:\pmdf\include\pmdf_err.h (NT).

Table 22–6 Completion Statuses

Status	Description
1	Success. Continues processing the message.

Table 22–6 (Cont.) Completion Statuses

Status	Description
PMDF__FORCEBITBUCKET	Sends the entire message to the bitbucket channel. Similar to PMDF__FORCEDISCARD, but always goes to the bitbucket channel and never to the filter_discard channel.
PMDF__FORCEDELETE	Deletes the current message part.
PMDF__FORCEDISCARD	Deletes the entire message. Either goes to the bitbucket channel or the filter_discard channel, depending on your configuration.
PMDF__FORCEHOLD	Holds the message as a .HELD file.
PMDF__FORCERETURN	Bounces the message.
PMDF__NOCHANGE	Tells the conversion channel that there were no changes made to the current message part.

These statuses can be used, for example, to discontinue processing the message when the command procedure or script determines that the message part contains objectionable content.

The following sections provide more details about each of these options.

22.1.4.1 Bouncing Messages

The conversion command procedure or script may tell PMDF to return the message to its sender, by using the PMDF__FORCERETURN completion status.

The command procedure or script may optionally use OUTPUT_DIAGNOSTIC to specify a text string to be included in the bounce message returned to the message sender.

On UNIX and NT, the script should write a line in the OUTPUT_OPTIONS file like:

```
OUTPUT_DIAGNOSTIC=text-string
```

On OpenVMS, the command procedure should define a logical name in the job logical name table like:

```
$ DEFINE/JOB OUTPUT_DIAGNOSTIC text-string
```

There are a couple of variations on the FORCERETURN behavior available.

A value of PMDF__FORCERETURN+1 causes PMDF to return not the original message text, but the final message text including any modifications done by the command procedure or script. The number of lines of the message returned is still controlled by the LINES_TO_RETURN option.

A value of PMDF__FORCERETURN-1 causes PMDF to return the entire message, including any modifications done by the command procedure or script.

22.1.4.2 Deleting Messages

The conversion command procedure or script may tell PMDF to delete the entire message, by using the PMDF__FORCEDISCARD or PMDF__FORCEBITBUCKET completion status.

The PMDF__FORCEDISCARD status causes PMDF to perform the same actions as for the Sieve discard command in mailbox filter files. For example, if the FILTER_DISCARD option is set to 2 or 3 in the PMDF option file, the message will be routed to the filter_discard channel (see Section 7.3.3), otherwise it will be routed to the bitbucket channel, where it will be immediately deleted.

The PMDF__FORCEBITBUCKET status causes PMDF to delete the message by always sending it to the bitbucket channel.

The command procedure or script may optionally use OUTPUT_DIAGNOSTIC to specify a text string to appear in the MAIL.LOG_CURRENT file.

22.1.4.3 Deleting Message Parts

The conversion command procedure or script may tell PMDF to delete the current message part, by using the PMDF__FORCEDELETE completion status.

This causes PMDF to perform the same action as for the DELETE=1 conversion parameter clause, however that clause deletes the part unconditionally.

22.1.4.4 Holding Messages

The conversion command procedure or script may tell PMDF to hold the message, by using the PMDF__FORCEHOLD completion status.

This causes PMDF to hold (sideline) the message as a .HELD file in the conversion channel queue.

22.1.4.5 No Changes

The command procedure or script may tell the conversion channel that it made no changes to the current message part by using the PMDF__NOCHANGE completion status. If **all** message parts get the PMDF__NOCHANGE status, then the conversion channel will forward the message on to the next channel unchanged (that is, the MIME boundaries and headers will remain unchanged, and any encoded message parts will be their original versions rather than the decoded and re-encoded versions). If any parts return anything other than PMDF__NOCHANGE, then the message is processed by the conversion channel as normal.

22.1.5 An Example on OpenVMS

The following are examples of

- a CONVERSIONS mapping table in the mappings file,
- conversions file entries, and
- a command procedure for the conversion channel to run.

These examples are derived from PMDF_ROOT:[DOC.EXAMPLES]VIRUS_SCAN.DCL.

Example 22–1 Sample CONVERSIONS Mapping

```
!
! CONVERSIONS mapping table
! A Yes enables the conversion channel, a No disables it.
!
! This example enables the conversion channel for internet mail destined for
! local VMS MAIL and MessageStore users, and other internel network users.
!
CONVERSIONS
  IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT              Yes
  IN-CHAN=tcp_local;OUT-CHAN=msgstore;CONVERT       Yes
  IN-CHAN=tcp_local;OUT-CHAN=tcp_internal;CONVERT   Yes
  IN-CHAN=*;OUT-CHAN=*;CONVERT                      No
```

Example 22–2 Sample Conversion Rule

```
!
! Perform virus scanning action on all APPLICATION/* message parts.
!
in-channel=tcp_local; in-type=application; in-subtype=*;
  parameter-symbol-0=NAME; parameter-copy-0=*;
  dparameter-copy-0=*; message-header-file=2;
  original-header-file=1; override-header-file=1;
  command="@pmdf_com:virus-scan.com"
```

Example 22–3 Sample Conversion Command Procedure

Example 22–3 Cont'd on next page

Example 22–3 (Cont.) Sample Conversion Command Procedure

```
$!-----
$! Main Processing
$!-----
$!
$! Generate unique filename
$ run PMDF_EXE:UNIQUE_ID
$ REFERENCE_ID = unique_id
$ RECORD_DIR = "put-the-path-to-the-record-dir-here"
$ RECORD_FILE = "’’RECORD_DIR’" + "’’REFERENCE_ID’"
$!
$! Run the virus scanner
$ call LAUNCH your-chosen-method
$!
$ if RED_FLAG .eq. 1
$ then
$     ! Virus detected, choose one of:
$     !#!call SUBSTITUTE_PART       ! Substitute a text part for the original
$     !#!call FORCE_HOLD            ! Force message to become .HELD
$     !#!call FORCE_BOUNCE          ! Force message to be bounced
$     !#!call FORCE_DELETE          ! Force offending part to be deleted
$     !#!call FORCE_DISCARD         ! Force the entire message to be deleted
$ else
$     ! No virus, just pass it on
$     copy ’INPUT_FILE’ ’OUTPUT_FILE’
$ endif
$!
$ EXIT
$!
$!-----
$! LAUNCH subroutine
$!
$! Usage: $ call LAUNCH METHOD_NAME
$! Returns: The symbol RED_FLAG with a value of either 0 or 1
$!-----
$!
$ LAUNCH: subroutine
$!
$ SCAN_METHOD = "’’P1’"
$ RED_FLAG == 0         ! Setup a global signalling flag
$!
$! VSWEEP virus scanner method
$ if SCAN_METHOD .eqs. "RUN_VSWEEP"
$ then
$    CONVERTER_COMMAND :== "command-to-invoke-VSWEEP-here"
$    CONVERTER_COMMAND ’INPUT_FILE’/ff/ns/il/output=’RECORD_FILE’.scan
$    if SWEEP$_STATUS .eqs. "SWEEP$_VIRUS" then RED_FLAG == 1
$ endif
$!
$! Template for other methods
$ if SCAN_METHOD .eqs. "RUN_METHOD_X"
$ then
$     CONVERTER_COMMAND :== "command-to-invoke-Method-X"
```

Example 22–3 Cont’d on next page

Example 22–3 (Cont.) Sample Conversion Command Procedure

```
$     CONVERTER_COMMAND 'INPUT_FILE'/output='RECORD_FILE'.scan
$     if METHOD_X_STATUS .eqs. "some-status-YY" then RED_FLAG == 1
$!
$ endif
$!
$ endsubroutine ! end of LAUNCH
$!
$!-----
$! Subroutines to set the completion status
$!-----
$!
$!--
$! Force the message to be held
$!--
$ FORCE_HOLD:  subroutine
$!
$ exit %x0A9C86AA   ! PMDF__FORCEHOLD from pmdf_com:pmdf_err.h
$!
$ endsubroutine
$!
$!--
$! Force the message part to be deleted
$!--
$!
$ FORCE_DELETE: subroutine
$!
$ exit %x0A9C8662   ! PMDF__FORCEDELETE from pmdf_com:pmdf_err.h
$!
$ endsubroutine
$!
$!--
$! Force the entire message to be deleted
$!--
$!
$ FORCE_DISCARD: subroutine
$!
$ exit %x0A9C86B3   ! PMDF__FORCEDISCARD from pmdf_com:pmdf_err.h
$!
$!--
$! Force a bounce.
$! parameter can be SWAP_PAYLOAD or SWAP_PAYLOAD_FULL
$!--
$!
$ FORCE_BOUNCE: subroutine
$!
$ if "''P1'" .eqs. "SWAP_PAYLOAD_FULL"
$ then
$     exit %x0A9C8579       ! PMDF__FORCEBOUNCE - 1
$ else if "''P1'" .eqs. "SWAP_PAYLOAD"
$ then
$     exit %x0A9C857B       ! PMDF__FORCEBOUNCE + 1
$ else
$     exit %x0A9C857A       ! PMDF__FORCEBOUNCE from pmdf_com:pmdf_err.h
```

Example 22–3 Cont'd on next page

Example 22–3 (Cont.) Sample Conversion Command Procedure

```
$ endif
$ endif
$!
$ endsubroutine
$!
$!-----
$! Generate some text to substitute for the part before delivering it.
$!-----
$!
$ SUBSTITUTE_PART: subroutine
$!
$! Insert a note saying that there has been a substitution.
$  open/write q_fh 'OUTPUT_FILE
$  write q_fh ""
$  write q_fh " The original document has been removed from this message"
$  write q_fh " The document was removed because .... "
$  write q_fh " The name of the original document was ''NAME'."
$  write q_fh ""
$  write q_fh "                          Enterprise Messaging Team"
$  write q_fh "                          Example.com Mail Services."
$  write q_fh ""
$  close q_fh
$!
$! Change the output mode to TEXT and the output encoding to NONE
$  define/job OUTPUT_MODE TEXT
$  define/job OUTPUT_ENCODING NONE
$!
$! Change the MIME information of this current message part to
$! TEXT/PLAIN and change the document name  references.
$  open/write fh 'OUTPUT_HEADERS
$  write fh "Content-type: TEXT/PLAIN; NAME=Substitute.txt"
$  write fh "Content-description: Alert about possible virus"
$  write fh "Content-disposition: attachment; filename=Substitute.txt"
$  write fh "Content-transfer-encoding: 7bit "
$  write fh ""
$  close fh
$!
$ endsubroutine
$!
```

22.2 Script Channel

The `script` channel is similar to the `conversion` channel in that arbitrary processing can be done on messages flowing through PMDF. The difference is that the `script` channel passes the entire message to the command procedure or script, instead of body part by body part. As with the `conversion` channel, any set of programs or command procedures can be used to perform script channel processing. For instance, third party virus or spam scanning software may be hooked in for automatic execution via the `script` channel. Or sites may develop their own custom applications to hook in.

22.2.1 Selecting Traffic for Processing

Although script channel processing is done using a regular PMDF channel program, under normal circumstances this channel is never specified directly either in an address or in a PMDF rewrite rule. PMDF controls access to the `script` channel via the `SCRIPT` mapping table in the PMDF mappings file.

As PMDF processes each message it probes the `SCRIPT` mapping (if one is present) with a string of the form

```
IN-CHAN=source-channel;OUT-CHAN=destination-channel;SCRIPT
```

where *source-channel* is the source channel from which the message is coming and *destination-channel* is the destination channel to which the message is heading. If a match occurs the resulting string should be a comma-separated list of keywords. Table 22–7 lists the available keywords.

Note: Make sure that there is no whitespace in the resulting string, for example around commas or equal signs.

Table 22–7 SCRIPT Mapping Table Keywords

Keyword	Action
Channel=*channel*	Enables script channel processing using the script channel named *channel*. Note that technically, the channel specified can be any channel defined in `pmdf.cnf`.
Maxblocks=*n*	Checks the size of the message. If it is larger than *n* blocks, then script channel processing is disabled, otherwise it is enabled.
Maxlines=*n*	Checks the number of lines in the message. If it is greater than *n*, then script channel processing is disabled, otherwise it is enabled.
No	Disables script channel processing.
Yes	Enables script channel processing.

A `No` is assumed if no match occurs.

If the `SCRIPT` mapping table enables the script channel, PMDF diverts the message from its regular destination to the script channel. If the script channel is not enabled, the message is queued to its regular destination channel.

For example, suppose messages require script processing if they come from outside your organization and are destined for either local users or remote MAIL-11 (DECnet) users. The following mapping would then be appropriate:

```
SCRIPT

  IN-CHAN=tcp_local;OUT-CHAN=l;SCRIPT          Yes
  IN-CHAN=tcp_local;OUT-CHAN=d;SCRIPT          Yes
  IN-CHAN=*;OUT-CHAN=*;SCRIPT                  No
```

Note: The SCRIPT mapping table is not checked for messages which have already been discarded, for example by a mailbox filter.

22.2.2 Script Channel Definition and Rewrite Rules

The first step is to add the script channel to the PMDF configuration file. The entry should have the form:

```
script
SCRIPT-DAEMON
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the script channel. Something like

```
script                    $U%script.localhostname@SCRIPT-DAEMON
script.localhostname      $U%script.localhostname@SCRIPT-DAEMON
```

where *localhostname* is the name of the local PMDF system, provides the necessary functionality. Once this is done, addresses of the form

user%host@script.*localhostname*

are routed through the script channel regardless of what the SCRIPT mapping says.

Additional script_* channels may be defined, for example to allow for different command procedures or scripts to be run for different source or destination channels. For example:

```
script_2
SCRIPT2-DAEMON
```

These alternate script channels are generally selected using the SCRIPT mapping table Channel keyword (see Table 22–7). If desired, you may also create separate rewrite rules for each of these alternate script channels, similar to the ones shown above.

Note: A channel named script **must** be defined to enable script channel processing. This channel is used as the default script channel.

22.2.3 Script Channel Option File

The next step is to create a script channel option file. The name of the option file is *x*_option where *x* is the name of the channel, hence usually script_option, and the file should be placed in the PMDF table directory.

This file is used to tell the script channel what command to run, using the COMMAND option. For example, to run a site-supplied virus checking command procedure on OpenVMS:

```
COMMAND=@pmdf_com:check-virus.com/output=pmdf_log:check-virus.out
```

The default command run by the script channel if none is supplied in an option file is @pmdf_com:script_command.com on OpenVMS, /pmdf/bin/script_script.sh on UNIX, and C:\pmdf\exe\script_batch.bat on NT. Note that there is no default version of this file supplied by PMDF.

22.2.4 Input and Output Symbols

Table 22–8 lists the items which are defined by the `script` channel to pass information to the `script` command. On OpenVMS, these are defined as DCL symbols in the subprocess that runs the `script` command. On UNIX and NT, these are defined as environment variables.

Table 22–8 Script Channel Input Symbols

	Name	Description
	ENVELOPE_FROM	The envelope From: address of the message.
	ENVELOPE_TO_FILE	The name of the file containing the envelope To: addresses of the message. Multiple addresses are included in the file one per line.
	INPUT_FILE	The name of the file containing the original message, including both headers and body. The command should read this file.
	ORIG_ENV_TO_FILE	The name of the file containing the original envelope To: addresses of the message. Multiple addresses are included in the file one per line.
	OUTPUT_FILE	The name of the file where the command should store its output. The command should create this file and write to it the modified version of the message, including both headers and body.
†	OUTPUT_OPTIONS	The name of the file to which the command should write options (such as status values).

†Available on UNIX and NT only

Note: The envelope From and envelope To information is **read-only**. The command procedure or script cannot modify them.

Table 22–9 lists the items which can be set by the `script` command to pass information back to the `script` channel. On OpenVMS, the command procedure should set them as logical names in the job logical name table, using the DCL command DEFINE/JOB. On UNIX or NT, the script the script should write them as options in the `OUTPUT_OPTIONS` file.

Table 22–9 Script Channel Output Symbols

	Name	Description
	OUTPUT_DIAGNOSTIC	Text to include in the error text returned to the message sender if a message is forcibly bounced via the `PMDF__FORCERETURN` completion status.
†	STATUS	The PMDF completion status for the script.

†Available on UNIX and NT only

22.2.5 Command Completion Statuses

The command procedure or script specified in the script channel option file COMMAND option should return one of the following completion statuses. On OpenVMS, this is accomplished by exiting from the command procedure with the desired status. On UNIX and NT, this is accomplished by setting the STATUS option in the OUTPUT_OPTIONS file. The script itself should exit explicitly with a value of 0, otherwise the script channel will think that the running of the script failed.

Table 22–10 shows the list of completion statuses that can be specified. The values for these statuses are defined in pmdf_com:pmdf_err.h (OpenVMS), /pmdf/include/pmdf_err.h (UNIX), or C:\pmdf\include\pmdf_err.h (NT).

Table 22–10 Completion Statuses

Status	Description
1	Success. Continues processing the message.
PMDF__FORCEBITBUCKET	Sends the entire message to the bitbucket channel. Similar to PMDF__FORCEDISCARD, but always goes to the bitbucket channel and never to the filter_discard channel.
PMDF__FORCEDISCARD	Deletes the entire message. Either goes to the bitbucket channel or the filter_discard channel, depending on your configuration.
PMDF__FORCEHOLD	Holds the message as a .HELD file.
PMDF__FORCERETURN	Bounces the message.
PMDF__NOCHANGE	Tells the script channel that there were no changes made to the message.

These statuses can be used, for example, to discontinue processing the message when the command procedure or script determines that the message contains objectionable content.

The following sections provide more details about each of these options.

22.2.5.1 Bouncing Messages

The script channel command procedure or script may tell PMDF to return the message to its sender, by using the PMDF__FORCERETURN completion status.

The command procedure or script may optionally use OUTPUT_DIAGNOSTIC to specify a text string to be included in the bounce message returned to the message sender.

On UNIX and NT, the script should write a line in the OUTPUT_OPTIONS file like:

```
OUTPUT_DIAGNOSTIC=text-string
```

On OpenVMS, the command procedure should define a logical name in the job logical name table like:

```
$ DEFINE/JOB OUTPUT_DIAGNOSTIC text-string
```

Note that using the base PMDF__FORCERETURN status value, only a sample of the original message is included in the bounce message. The amount is controlled by the LINES_TO_RETURN PMDF option (see Section 7.3.4). However, there are a couple of variations on the FORCERETURN behavior available.

A value of PMDF__FORCERETURN+1 causes PMDF to return not the original message text, but the final message text including any modifications done by the command procedure or script. The number of lines of the message returned is still controlled by the LINES_TO_RETURN option.

A value of PMDF__FORCERETURN-1 causes PMDF to return the entire message, including any modifications done by the command procedure or script.

22.2.5.2 Deleting Messages

The script channel command procedure or script may tell PMDF to delete the message, by using the PMDF__FORCEDISCARD or PMDF__FORCEBITBUCKET completion status.

The PMDF__FORCEDISCARD status causes PMDF to perform the same actions as for the Sieve discard command in mailbox filter files. For example, if the FILTER_DISCARD option is set to 2 or 3 in the PMDF option file, the message will be routed to the filter_discard channel (see Section 7.3.3), otherwise it will be routed to the bitbucket channel, where it will be immediately deleted.

The PMDF__FORCEBITBUCKET status causes PMDF to delete the message by always sending it to the bitbucket channel.

The command procedure or script may optionally use OUTPUT_DIAGNOSTIC to specify a text string to appear in the MAIL.LOG_CURRENT file.

22.2.5.3 Holding Messages

The script channel command procedure or script may tell PMDF to hold the message, by using the PMDF__FORCEHOLD completion status.

This causes PMDF to hold (sideline) the message as a .HELD file in the script channel queue.

22.2.5.4 No Changes

If the command procedure or script returns PMDF__NOCHANGE, this lets the script channel know that the command procedure made no changes to the message. The script channel will forward the message on to the next channel completely unchanged.

22.2.6 Using Multiple Script Channels

Multiple script channels can be configured. Separate script channels can be used to run different command procedures or scripts.

First, multiple script channels must be defined in the PMDF configuration file. For example:

```
script
SCRIPT-DAEMON

script_msgstore
SCRIPTMS-DAEMON
```

Note: Make sure you define a channel named exactly `script`.

Second, create channel option files for each of these channels, containing the command to execute. Using the example script channels above (for UNIX), create a `/pmdf/table/script_option` file for the `script` channel, containing for example:

```
COMMAND=/pmdf/tmp/script.sh
```

Then create a `/pmdf/table/script_msgstore_option` file for the `script_msgstore` channel, containing for example:

```
COMMAND=/pmdf/tmp/script_msgstore.sh
```

Where `script.sh` and `script_msgstore.sh` are site-supplied scripts.

Third, add a `SCRIPT` mapping table, using the `channel` keyword, for example:

```
SCRIPT

  IN-CHAN=tcp_*;OUT-CHAN=l;SCRIPT           channel=script
  IN-CHAN=tcp_*;OUT-CHAN=msgstore;SCRIPT    channel=script_msgstore
  IN-CHAN=*;OUT-CHAN=*;SCRIPT               No
```

In this example, the `script.sh` script is run for messages destined for local UNIX mail delivery, and the `script_msgstore.sh` script is run for messages destined for MessageStore accounts.

22.3 Disclaimer Channel

The `disclaimer` channel is similar to the `script` and `conversion` channels in that it is an intermediate channel which makes modifications to messages as they pass through the channel on their way to their final destinations. In particular, the `disclaimer` channel adds selected text (usually in the form of a disclaimer) to the top, bottom, or headers of a message.

22.3.1 Selecting Traffic for Processing

Although disclaimer channel processing is done using a regular PMDF channel program, under normal circumstances this channel is never specified directly either in an address or in a PMDF rewrite rule. PMDF controls access to the `disclaimer` channel via the `DISCLAIMER` mapping table in the PMDF mappings file.

As PMDF processes each message it probes the `DISCLAIMER` mapping (if one is present) with a string of the form

```
IN-CHAN=source-channel;OUT-CHAN=destination-channel;DISCLAIMER
```

where *source-channel* is the source channel from which the message is coming and *destination-channel* is the destination channel to which the message is heading. If a match occurs the resulting string should be a comma-separated list of keywords. Table 22–11 lists the available keywords.

Note: Make sure that there is no whitespace in the resulting string, for example around commas or equal signs.

Table 22–11 DISCLAIMER Mapping Table Keywords

Keyword	Action
Channel=*channel*	Enables disclaimer channel processing using the disclaimer channel named *channel*.
No	Disables disclaimer channel processing.
Yes	Enables disclaimer channel processing.

A `No` is assumed if no match occurs.

If the `DISCLAIMER` mapping table enables the disclaimer channel, PMDF diverts the message from its regular destination to the disclaimer channel. If the disclaimer channel is not enabled, the message is queued to its regular destination channel.

For example, suppose your organization has mandated that all messages have disclaimers added to them if they originate from inside your company and are destined for the Internet. The following mapping would then be appropriate:

```
DISCLAIMER

  IN-CHAN=l;OUT-CHAN=tcp_local;DISCLAIMER  yes
  IN-CHAN=tcp_internal;OUT-CHAN=tcp_local;DISCLAIMER yes
  IN-CHAN=*;OUT-CHAN=*;DISCLAIMER   no
```

If you want to have only a subset of your users use the disclaimer channel, or if you want some users to use a different disclaimer channel than other users, you can redirect people to a different incoming channel using the `switchchannel` channel keyword and IP-address based rewrite rules. The `DISCLAIMER` mapping table can then be set up to recognize those channels as the incoming channel and turn on or off the disclaimer channel, or use an alternate disclaimer channel, as desired.

Note: The DISCLAIMER mapping table is not checked for messages which have already been discarded, for example by a mailbox filter.

22.3.2 Disclaimer Channel Definition and Rewrite Rules

The first step is to add the disclaimer channel to the PMDF configuration file. The entry should have the form:

```
disclaimer
DISCLAIMER-DAEMON
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the disclaimer channel. Something like

```
disclaimer                    $U%disclaimer.localhostname@DISCLAIMER-DAEMON
disclaimer.localhostname      $U%disclaimer.localhostname@DISCLAIMER-DAEMON
```

where *localhostname* is the name of the local PMDF system, provides the necessary functionality. Once this is done, addresses of the form

user%host@disclaimer.*localhostname*

are routed through the disclaimer channel regardless of what the DISCLAIMER mapping says.

Additional disclaimer_* channels may be defined, for example to allow for different disclaimer text to be added to messages coming from different source channels or headed for different destination channels. For example:

```
disclaimer_2
DISCLAIMER2-DAEMON
```

These alternate disclaimer channels are generally selected using the DISCLAIMER mapping table Channel keyword (see Table 22–11). If desired, you may also create separate rewrite rules for each of these alternate disclaimer channels, similar to the ones shown above.

Note: A channel named disclaimer **must** be defined to enable disclaimer channel processing. This channel is used as the default disclaimer channel.

22.3.3 Disclaimer Channel Option File

The next step is to create a disclaimer channel option file, if needed. The name of the option file is *x*_option where *x* is the name of the channel, hence usually disclaimer_option, and the file should be placed in the PMDF table directory.

The option file is used to tell the disclaimer channel what exact actions to perform and where to get the text to add to messages. There are five different places where the disclaimer channel can add text to a message:

- in the header using X-Disclaim headers
- to the top of plain text messages
- to the bottom of plain text messages

- to the top of HTML text messages
- to the bottom of HTML text messages

Note that the option file is not required. If no option file exists, the default operation is to append the text that is in the file `disclaimer.txt` in the PMDF table directory to the bottom of both plain text and HTML text messages.

By using an option file, you can instruct the disclaimer channel to add text at any combination of the above locations, using different files containing different text for each one. For example, this lets you specify a plain text disclaimer to be added to plain text messages, and text with HTML code to be added to HTML messages.

Similar to the `conversion` and `script` channels, you can define multiple `disclaimer` channels, and invoke different ones based on the `DISCLAIMER` mapping table. Each disclaimer channel has a different option file, in which you can specify different text. For example, you might want your sales department to have a different disclaimer than the engineering department.

22.3.3.1 Format of the Option File

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

option=value

value can be either a string or an integer, depending on the option's requirements.

22.3.3.2 Available Disclaimer Channel Options

The available options are:

DEFAULT_FILE (file-name)
The `DEFAULT_FILE` option specifies a different default file name or location for the default file to use. This option is not required. If it is not specified, the default file used is `pmdf_table:disclaimer.txt`.

HEADER (file-name or DEFAULT_FILE)
The `HEADER` option tells the disclaimer channel to add `X-Disclaim` headers to the message containing the text in the file specified. The value may also be the keyword `DEFAULT_FILE`, which indicates that the disclaimer channel should use the text in the default file. If this option is not specified, no headers are added. If the text to be added contains multiple lines, by default a single `X-Disclaim` header is added containing all of the text, with continuation lines if necessary. The `MULTIPLE_HEADERS` option may be specified with a value of 1 to indicate that instead multiple `X-Disclaim` headers should be added, one for each line of text.

HTML_BOTTOM (file-name or DEFAULT_FILE)
The `HTML_BOTTOM` option tells the disclaimer channel to add the text in the file specified to the bottom of `text/html` messages (or message parts of multipart messages). The value may also be the keyword `DEFAULT_FILE`, which indicates that the disclaimer channel should use the text in the default file. If this option is not specified, no text is appended to `text/html` messages.

HTML_TOP (file-name or DEFAULT_FILE)

The HTML_TOP option tells the disclaimer channel to add the text in the file specified to the top of text/html messages (or message parts of multipart messages). The value may also be the keyword DEFAULT_FILE, which indicates that the disclaimer channel should use the text in the default file. If this option is not specified, no text is prepended to text/html messages.

MULTIPLE_HEADERS (0 or 1)

The MULTIPLE_HEADERS option modifies the action of the disclaimer channel when adding text to messages as X-Disclaim headers. The default value is 0, meaning that only one X-Disclaim header is added to messages, containing the entire text, using continuation lines if necessary. If the value is 1, and the text to be added spans multiple lines, then multiple X-Disclaim headers are added, one for each line of text.

PLAIN_BOTTOM (file-name or DEFAULT_FILE)

The PLAIN_BOTTOM option tells the disclaimer channel to add the text in the file specified to the bottom of text/plain messages (or message parts of multipart messages). The value may also be the keyword DEFAULT_FILE, which indicates that the disclaimer channel should use the text in the default file. If this option is not specified, no text is appended to text/plain messages.

PLAIN_TOP (file-name or DEFAULT_FILE)

The PLAIN_TOP option tells the disclaimer channel to add the text in the file specified to the top of text/plain messages (or message parts of multipart messages). The value may also be the keyword DEFAULT_FILE, which indicates that the disclaimer channel should use the text in the default file. If this option is not specified, no text is prepended to text/plain messages.

22.3.3.3 Example Disclaimer Channel Option File

Here's an example option file:

```
DEFAULT_FILE=/myfiles/disclaimer/default.txt
HEADER=/myfiles/disclaimer/header.txt
MULTIPLE_HEADERS=0
PLAIN_TOP=DEFAULT_FILE
HTML_TOP=DEFAULT_FILE
PLAIN_BOTTOM=/myfiles/disclaimer/plain/bottom.txt
HTML_BOTTOM=/myfiles/disclaimer/html/bottom.htm
```

22.3.4 Files Containing Disclaimer Text

You have to create the files containing your disclaimer text yourself. By default, this would be the file pmdf_table:disclaimer.txt. You also have to create the files pointed to by any options you specified in the options file.

For text to be added to the headers (specified by the HEADER option) we suggest that it is short and confined to a single line. The total length must be no more than 1024 characters. It can have multiple lines, although that is not recommended. By default, any line breaks are removed and the entire text is added as a single X-Disclaim header. However, if you specify MULTIPLE_HEADERS=1 in the option file, each line is added as a separate X-Disclaim header. Note that you do not need to specify the header label

to use inside your disclaimer text. The header label X-Disclaim is always added by PMDF.

The files for the plain and HTML top and bottom text can be any length and contain any text you'd like. For example, the ones intended to be added to HTML messages can contain HTML code such as links or images.

22.3.5 Message Integrity Issues

The disclaimer channel takes great pains to make sure that it does not modify messages in such a way that it compromises the integrity of the message. For example, it does not make any modifications to the bodies of any signed messages (which would invalidate the signature), or reports such as NOTARY messages or NDNs (Non-Delivery Notices).

For simple (non-multipart) messages, the disclaimer channel only adds disclaimers to the bodies of text/plain and text/html messages. Messages of any other non-multipart content-type are not modified.

For multipart messages, the disclaimer channel only looks at the following content-types:

```
multipart/alternative
multipart/mixed
multipart/related
multipart/parallel
```

Within the multipart message, only parts that are text/plain or text/html are modified. Only the first part of each type is modified.

For each text/plain or text/html part or entire message, additional checks are made to make sure it is a part or message that should be modified. For example, this ensures that PMDF does not modify attachments, such as files or images or binary executables.

The disclaimer channel also makes sure that it does not add the disclaimers twice by marking each message that it modifies with a X-Disclaim-Comment header.

22.4 Running More Than One Of These Channels

Since PMDF checks all of the CONVERSIONS, SCRIPT, and DISCLAIMER mapping tables at the same time, it can happen that two or all three of the conversion, script, and disclaimer channels are enabled. Since PMDF can queue a message to only one of these channels, it chooses the channel whose mapping table appears first in the mappings file.

Two or more of the conversion, script, and disclaimer channels can all be run, one after the other, by setting up the channels' mapping tables correctly.

For example, the following mapping tables cause the `script` channel to run first, followed by the `conversion` channel.

```
SCRIPT

  IN-CHAN=tcp_local;OUT-CHAN=l;SCRIPT           Yes
  IN-CHAN=tcp_local;OUT-CHAN=msgstore;SCRIPT    Yes
  IN-CHAN=*;OUT-CHAN=*;SCRIPT                   No

CONVERSIONS

  IN-CHAN=script;OUT-CHAN=l;CONVERT             Yes
  IN-CHAN=script;OUT-CHAN=msgstore;CONVERT      Yes
  IN-CHAN=*;OUT-CHAN=*;CONVERT                  No
```

23 BSMTP Channels: MTA to MTA Tunnelling

This chapter describes how to tunnel messages between two or more cooperating PMDF systems using Batch SMTP (BSMTP). In addition, use of PMDF's general conversion facilities to provide services such as payload compression and digital signatures for authentication and integrity is described.

Batch SMTP (BSMTP) is a batch-mode implementation of the SMTP protocol which turns SMTP into a remote-submission protocol. For over a decade, batch SMTP was used quite heavily as a message transfer protocol on the international BITNET network. Cooperating PMDF sites can use BSMTP as an effective means of moving mail in bulk between one another; for instance the exchange of company e-mail between two company offices by means of the Internet.

With BSMTP, messages are bundled together on one PMDF system and then periodically transmitted through arbitrary MTAs and networks to a remote PMDF system. Upon receipt at the remote system, the bundle is unpacked and the individual messages sent on to their recipients. With PMDF's general conversion facilities, arbitrary transformations can be performed on the bundles such as document conversion, compression, addition of digital signatures for authentication and integrity, *etc.* This chapter provides examples of compression using the Free Software Foundation's `GZIP` and `GUNZIP` utilities and authentication using Pretty Good Privacy, Inc.'s PGP® utility.[1]

23.1 Configuring the BSMTP Channels

Each of the PMDF systems which will be exchanging mail via `BSMTP` will need one incoming `BSMTP` channel and an outgoing `BSMTP` channel for each of the remote PMDF systems. The channel definitions should be along the lines of:

```
bsin_gateway smtp
bsin.host0

bsout_remote1 smtp master user bsmtp daemon host1
BSOUT-REMOTE1

bsout_remote2 smtp master user bsmtp daemon host2
BSOUT-REMOTE2

...

bsout_remoteN smtp master user bsmtp daemon hostN
BSOUT-REMOTEN
```

where *host0* is the name of the local PMDF host, as used by the other remote PMDF systems, and *host1*, *host2*, ..., *hostN* are the host names of the remote PMDF systems.

[1] Use of PGP for commercial purposes requires a license from Pretty Good Privacy, Inc. Please contact Pretty Good Privacy, Inc. for details and assistance in licensing PGP.

The strings *remote1*, *remote2*, ... *remoteN* and *REMOTE1*, *REMOTE2*, ..., *REMOTEN* are arbitrary and need just be distinct from one another.

With the above definitions, the channel bsout_ *remote1* will bundle up its BSMTP parcels and send them on to the fixed address bsmtp@*host1*. Likewise for the remaining BSOUT channels.

The rewrite rules appear as

```
domain1      $U%$H@BSOUT-REMOTE1$Nbsout_remote1
.domain1     $U%$H$D@BSOUT-REMOTE1$Nbsout_remote1
domain2      $U%$H@BSOUT-REMOTE2$Nbsout_remote2
.domain2     $U%$H$D@BSOUT-REMOTE2$Nbsout_remote2
...
domainN      $U%$H@BSOUT-REMOTEN$Nbsout_remoteN
.domainN     $U%$H$D@BSOUT-REMOTEN$Nbsout_remoteN
```

where *domain1*, *domain2*, ... *domainN* are the domain names of the remote PMDF systems.

Finally, add to the FORWARD mapping table the entry

```
FORWARD

  bsmtp@host0    bsmtp@bsin.host0$Y$D
```

where, again, *host0* is the host name for the local PMDF system which will be used by the BSOUT channels on the remote PMDF systems. That way, when they send BSMTP parcels to bsmtp@*host0*, it will be forwarded on to the local bsin_gateway channel.[2]

For example, assume that the example.com domain will be exchanging BSMTP traffic with the example.co.uk domain via the PMDF hosts hub.example.com and athena.example.co.uk. Then hub.example.com would have the configuration

```
example.co.uk    $U%$H@BSOUT-REMOTE1$Nbsout_remote1
.example.co.uk   $U%$H$D@BSOUT-REMOTE1$Nbsout_remote1

...

bsin_gateway smtp
bsin.hub.example.com

bsout_remote1 smtp master user bsmtp daemon athena.example.co.uk
BSOUT-REMOTE1
```

and the FORWARD mapping table entry

```
FORWARD

  bsmtp@hub.example.com  bsmtp@bsin.hub.example.com$Y$D
```

[2] Any of several mechanisms might be used to accomplish this forwarding. The most efficient is the use of an alias when *host0* is the official local host name for the PMDF system. The least efficient is the FORWARD mapping table; which method is best for a given site depends upon site-specific issues. Use of the FORWARD mapping table is presented here because that method works in all cases.

The system athena.example.co.uk would have the configuration

```
example.com     $U%$H@BSOUT-REMOTE1$Nbsout_remote1
.example.com    $U%$H$D@BSOUT-REMOTE1$Nbsout_remote1

...

bsin_gateway smtp
bsin.athena.example.co.uk

bsout_remote1 smtp master user bsmtp daemon hub.example.com
BSOUT-REMOTE1
```

and the FORWARD mapping table entry

```
FORWARD

  bsmtp@athena.example.co.uk  bsmtp@bsin.athena.example.co.uk$Y$D
```

With the above configurations, when a user on hub.example.com sends mail to user@example.co.uk, the message is routed to the bsout_remote1 channel. That channel will package the message up into a BSMTP parcel and send that parcel on to bsmtp@athena.example.co.uk. Owing to the $Nbsout_remote1 tag in the example.co.uk rewrite rules, those rewrite rules will be ignored when the bsout_remote1 channel enqueues the message. Instead, the normal rewrite rules for example.co.uk will take effect and route the message containing the parcel out to the WAN (*e.g.,* the Internet).

Note that the outbound BSMTP channels can construct application/batch-smtp message parts containing multiple messages. As such, sites may want to use the after channel keyword on their BSOUT channels. So doing may prove advantageous for sites who want to bundle their mail up into large parcels and send those parcels only once every few minutes, hours, or days. Also, the ATTEMPT_TRANSACTIONS_PER_SESSION channel option might be used with the BSOUT channels to prevent cases where, under heavy load, a BSOUT channel just runs continuously bundling into a single parcel messages queuing up to be sent out. This option puts an upper limit on the number of messages placed in a single parcel and forces the channel to close a parcel, send it along, and start a new parcel when there are lots of messages to bundle up.

23.2 Performing the Desired Message Transformation via Service Conversions

PMDF's service conversion facility, described in Section 6.4, may be used with BSMTP channels to perform desired message transformations on incoming and outgoing messages.

Usually outgoing BSMTP channels, BSOUT channels, are configured to perform one sort of service conversion on the messages they emit, and incoming BSMTP channels, BSIN channels, are configured to perform the inverse service conversion on messages they receive. Thus when BSMTP channels are used, the PMDF mapping file would usually contain a CHARSET-CONVERSION mapping such as:

```
CHARSET-CONVERSION

  in-chan=bsout_*;out-chan=*;convert      yes
  in-chan=*;out-chan=bsin_*;convert       yes
```

Note that the CHARSET-CONVERSION entries shown are such as to enable service conversions for messages sent from BSOUT channels (such as messages transitting through a BSOUT channel on their way out to an outgoing TCP/IP channel), as well as for messages sent to a BSIN channel (such as messages transitting through a BSIN channel on their way in from an incoming TCP/IP channel).

Once execution of service conversions has been enabled via a CHARSET-CONVERSION mapping such as that shown above, the specific service conversions to be performed must be configured in the PMDF conversions file. Section 23.2.2 provides examples of configuring specific service conversions on OpenVMS; Section 23.2.3 provides examples of configuring specific service conversions on UNIX. Note that the sample command procedures cited in the examples are available in the PMDF examples directory, PMDF_ROOT:[OLD-doc.examples] (OpenVMS) or /pmdf/root/doc/examples/ (UNIX).

23.2.1 BSOUT Channel Option Files

An option file may be used to control characteristics of BSOUT channels. Such an option file must be named *x*_option where *x* is the name of the channel, and stored in the PMDF table directory. For instance, if a channel is named bsout_host1, then its option file would be PMDF_TABLE:bsout_host1_option. on OpenVMS, or /pmdf/table/bsout_host1_option on UNIX, or (drive possibly varying with installation) C:\pmdf\table\bsout_host1_option on NT.

SMTP channel options, as described in Section 21.1.2, are available, but most are not of interest in the context of a BSMTP channel. The one available option which is likely to be of interest is:

ATTEMPT_TRANSACTIONS_PER_SESSION (integer)
Set a limit on the number of message files composed into one BSMTP message.

23.2.2 Examples on OpenVMS

The following subsections provide examples of using BSMTP channels on OpenVMS.

23.2.2.1 Configuring the BSMTP Channels to Compress Their Payloads on OpenVMS

Using PMDF's general purpose, on-the-fly conversion facilities, BSMTP parcels can be compressed on the sending system and then uncompressed on the receiving system. This allows for faster transmission of the parcels through the network.

In the CHARSET-CONVERSION mapping table on each PMDF system, a simple entry enabling conversions for the BSMTP channels must be made:

```
CHARSET-CONVERSION

  in-chan=bsout_*;out-chan=*;convert      yes
  in-chan=*;out-chan=bsin_*;convert       yes
```

In the PMDF conversions file on each system, conversion entries are added which call out to the site-supplied command procedure, PMDF_COM:compress.com:

```
in-chan=bsout_*; part-number=1; in-type=*; in-subtype=*;
  service-command="@PMDF_COM:COMPRESS.COM COMPRESS 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=bsin_*; part-number=1; in-type=application;
  in-subtype=compressed-bsmtp;
  service-command="@PMDF_COM:COMPRESS.COM DECOMPRESS 'INPUT_FILE' 'OUTPUT_FILE'"
```

The PMDF_COM:compress.com: command procedure is shown in Figure 23–1.

Figure 23–1 compress.com**: Compress and decompress BSMTP payloads**

```
$ !
$ ! Compress/decompress a MIME message using GZIP & GUNZIP
$ ! P1 == "COMPRESS" | "DECOMPRESS"
$ ! P2 == File to compress or decompress
$ ! P3 == File containing the compressed or decompressed result
$ !
$ ! Ensure that we have three command line arguments
$ IF P1 .EQS. "" THEN EXIT 229448  ! DCL-W-INSFPRM
$ IF P2 .EQS. "" THEN EXIT 229448
$ IF P3 .EQS. "" THEN EXIT 229448
$ !
$ ! Used for temporary files
$ OUTFILE = F$ELEMENT(0,";",P3)
$ !
$ ! Dispatch to the correct part of this command file
$ IF "DECOMPRESS" .EQS. F$EDIT(P1,"TRIM,UPCASE) THEN GOTO DECOMPRESS
$ IF "COMPRESS" .NES. F$EDIT(P1,"TRIM,UPCASE) THEN EXIT 229472  ! DCL-W-IVKEYW
$ !
$ COMPRESS:
$   GZIP = "$PMDF_EXE:GZIP.EXE"
$   DEFINE/USER SYS$OUTPUT 'OUTFILE'-TMP
$   GZIP -C 'P2'
$   PMDF ENCODE/HEADER/TYPE=APPLICATION/SUBTYPE=COMPRESSED-BSMTP -
      'OUTFILE'-TMP 'P3'
$   DELETE/NOLOG 'OUTFILE'-TMP;*
$   EXIT 1
```

Figure 23–1 Cont'd on next page

Figure 23–1 (Cont.) `compress.com`**: Compress and decompress BSMTP payloads**

```
$ !
$ DECOMPRESS:
$   GUNZIP = "$PMDF_EXE:GUNZIP.EXE"
$   PMDF DECODE/HEADER 'P2' 'OUTFILE'-TMP
$   DEFINE/USER SYS$OUTPUT 'P3'
$   GUNZIP -C 'OUTFILE-TMP'
$   DELETE/NOLOG 'OUTFILE'-TMP;*
$   EXIT 1
```

23.2.2.2 Configuring the BSMTP Channels to Provide Authentication Services on OpenVMS

Using PMDF's general purpose, on-the-fly conversion facilities, authentication and integrity services may be tied in to the BSMTP channels. This is done through the `CHARSET-CONVERSION` mapping table, the PMDF conversions file, and a site-supplied command procedure to digitally sign payloads and verify the signature and integrity of the data upon receipt.

In the `CHARSET-CONVERSION` mapping table on each PMDF system, a simple entry enabling conversions for the BSMTP channels must be made:

```
CHARSET-CONVERSION

  in-chan=bsout_*;out-chan=*;convert      yes
  in-chan=*;out-chan=bsin_*;convert       yes
```

In the PMDF conversions file on each system, there must be conversion entries to invoke the site-supplied command procedures:

```
in-chan=bsout_*; part-number=1; in-type=*; in-subtype=*;
  service-command="@PMDF_COM:PGP_SIGN.COM 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=bsin_*; part-number=1; in-type=multipart; in-subtype=signed;
  service-command="@PMDF_COM:PGP_VERIFY.COM 'INPUT_FILE' 'OUTPUT_FILE'"
```

These two command procedures are shown in Figures 23–2 and 23–3. They assume that the PGP utility is the image `D1:[pgp]pgp.exe`. Note that the `pgp_sign.com` procedure requires the pass phrase for the PMDF MTA's private PGP key in order to generate signatures. Edit the procedure to reflect the correct pass phrase and be sure to protect the file from other users:

```
$ SET FILE/OWNER=[PMDF] PMDF_COM:PGP_SIGN.COM
$ SET PROTECTION=(S:RWED,O:RWED,G,W) PMDF_COM:PGP_SIGN.COM
```

Figure 23–2 `pgp_sign.com`**: Digitally sign BSMTP payloads**

```
$ !   P1 == Input file specification; message to sign
$ !   P2 == Output file specification; multipart/signed message
$ !   P3 == File specification for the file of envelope recipient addresses
```

Figure 23–2 Cont'd on next page

Figure 23–2 (Cont.) `pgp_sign.com`**: Digitally sign BSMTP payloads**

```
$ !
$ ! Check that we have at least two command line parameters
$ IF P1 .EQS. "" THEN EXIT 229448  ! DCL-W_INSFPRM
$ IF P2 .EQS. "" THEN EXIT 229448
$ !
$ ! Basic definitions
$ PGP     = "$D1:[PGP]PGP.EXE"
$ PGPUSER = "PMDF MTA key"
$ PGPPATH = "PMDF_ROOT:[TABLE.PGP]"
$ PGPPASS = "Percy eats pealed banannas"
$ FILENAM = F$ELEMENT(0,";",P2)
$ !
$ ! Error handling
$ ON ERROR THEN GOTO ERROR
$ ON SEVERE_ERROR THEN GOTO ERROR
$ !
$ ! Generate the digital signature
$ PGP "-sab" "-u" "''PGPUSER'" "-z" "''PGPPASS'" 'P1' "-o" 'FILENAM'-SIGN -
      "+batchmode"
$ !
$ ! Get a unique string to use in a MIME boundary marker
$ RUN PMDF_EXE:UNIQUE_ID.EXE
$ BOUNDARY = "''unique_id'"
$ !
$ ! Start the multipart message and the first message part
$ OPEN/WRITE/ERROR=ERROR OUTFILE 'P2'
$ WRT = "WRITE/ERROR=ERROR OUTFILE"
$ WRT "Content-type: multipart/signed; boundary=''BOUNDARY';"
$ WRT " micalg=pgp-md5; protocol=application/pgp-signature"
$ WRT ""
$ WRT "--''BOUNDARY'"
$ CLOSE/ERROR=ERROR OUTFILE
$ !
$ ! Start the second message part
$ OPEN/WRITE/ERROR=ERROR OUTFILE 'FILENAM'-MID
$ WRT "--''BOUNDARY'"
$ WRT "Content-type: application/pgp-signature"
$ WRT ""
$ CLOSE/ERROR=ERROR OUTFILE
$ !
$ ! And the end of the message
$ OPEN/WRITE/ERROR=ERROR OUTFILE 'FILENAM'-BOT
$ WRT "--''BOUNDARY'--"
$ CLOSE/ERROR=ERROR OUTFILE
$ !
$ ! Now glue all of the pieces together
$ CONVERT/APPEND 'P1' 'P2'
$ CONVERT/APPEND 'FILENAM'-MID 'P2'
$ CONVERT/APPEND 'FILENAM'-SIGN 'P2'
$ CONVERT/APPEND 'FILENAM'-BOT 'P2'
```

Figure 23–2 Cont'd on next page

Figure 23–2 (Cont.) `pgp_sign.com`**: Digitally sign BSMTP payloads**

```
$ !
$ ! Delete the temporary files
$ DELETE/NOLOG 'FILENAM'-MID;*,'FILENAM'-SIGN;*,'FILENAM'-BOT;*
$ EXIT 1
$ !
$ ! We fall through to here when we have an error
$ ERROR:
$ SET NOON
$ IF F$TRNLNM("OUTFILE") .NES. "" THEN CLOSE OUTFILE
$ DELETE/NOLOG 'FILENAM'-*.*,'P2'
$ SET ON
$ EXIT 2
```

Figure 23–3 `pgp_verify.com`**: Verify the integrity of a digitally signed BSMTP payload**

```
$ !   P1 == Input file specification; multipart/signed message
$ !   P2 == Output file specification; message which was signed;
$ !   P3 == File specification for the file of envelope recipient addresses
$ !
$ ! Check that we have at least two command line parameters
$ IF P1 .EQS. "" THEN EXIT 229448  ! DCL-W-INSFPRM
$ IF P2 .EQS. "" THEN EXIT 229448
$ !
$ ! Basic definitions
$ PGP     = "$D1:[PGP]PGP.EXE"
$ PGPPATH = "PMDF_ROOT:[TABLE.PGP]"
$ FILENAM = F$ELEMENT(0,";",P2)
$ !
$ ! Error handling
$ ON ERROR THEN GOTO ERROR
$ ON SEVERE_ERROR THEN GOTO ERROR
$ !
$ ! Reformat the input file to look like a PGP signature file
$ OPEN/READ/ERROR=ERROR INFILE 'P1'
$ OPEN/WRITE/ERROR=ERROR OUTFILE 'FILENAM'-SIGN
$ WRT = "WRITE/ERROR=ERROR OUTFILE"
$ STATE = 1
$ LOOP:
$   READ/ERROR=ERROR/END_OF_FILE=END_LOOP INFILE LINE
$   IF STATE .EQ. 1
$   THEN
$     IF F$EXTRACT(0,2,LINE) .EQS. "--"
$     THEN
$       STATE = 2
$       BOUNDARY = LINE
$       WRT "-----BEGIN PGP SIGNED MESSAGE-----"
$       WRT ""
$     ENDIF
```

Figure 23–3 Cont'd on next page

Figure 23–3 (Cont.) `pgp_verify.com`**: Verify the integrity of a digitally signed BSMTP payload**

```
$   ELSE
$     IF STATE .EQ. 2
$     THEN
$       IF BOUNDARY .NES. LINE
$       THEN
$         WRT LINE
$       ELSE
$         STATE = 3
$       ENDIF
$     ELSE
$       IF STATE .EQ. 3
$       THEN
$         IF LINE .EQS. ""
$         THEN
$           STATE = 4
$           WRT ""
$         ENDIF
$       ELSE
$         WRT LINE
$       ENDIF
$     ENDIF
$   ENDIF
$   GOTO LOOP
$ !
$ END_LOOP:
$ CLOSE/ERROR=ERROR INFILE
$ CLOSE/ERROR=ERROR OUTFILE
$ !
$ ! Now check the signature
$ DEFINE/USER SYS$OUTPUT 'FILENAM'-CHECK
$ PGP "-o" 'FILENAM'-OUT 'FILENAM'-SIGN "+batchmode"
$ !
$ ! See what the results of the check were; build the X-Content-MIC-check: line
$ SEARCH/OUTPUT='FILENAM'-MIC/EXACT 'FILENAM'-CHECK " signature from user "
$ IF $STATUS .EQ. 1
$ THEN
$   OPEN/READ/ERROR=ERROR INFILE 'FILENAM'-MIC
$   READ/ERROR=ERROR INFILE LINE
$   CLOSE/ERROR=ERROR INFILE
$   MIC_CHECK = "X-Content-MIC-check: "+LINE
$ ELSE
$   MIC_CHECK = "X-Content-MIC-check: Bad signature"
$ ENDIF
$ OPEN/WRITE/ERROR=ERROR OUTFILE 'P2'
$ WRITE/ERROR=ERROR OUTFILE MIC_CHECK
$ CLOSE/ERROR=ERROR OUTFILE
$ !
$ ! Now assemble the result: the MIC check + signed data
$ CONVERT/APPEND 'FILENAM'-OUT 'P2'
$ DELETE/NOLOG 'FILENAM'-*.*
$ EXIT 1
```

Figure 23–3 Cont'd on next page

Figure 23–3 (Cont.) `pgp_verify.com`**: Verify the integrity of a digitally signed BSMTP payload**

```
$ !
$ ! We fall through to here when there is an error
$ ERROR:
$ SET NOON
$ IF F$TRNLNM("INFILE") .NES. "" THEN CLOSE INFILE
$ IF F$TRNLNM("OUTFILE") .NES. "" THEN CLOSE OUTFILE
$ DELETE/NOLOG 'FILENAM'-*.*,'P2'
$ SET ON
$ EXIT 2
```

23.2.2.2.1 Using PGP with PMDF on OpenVMS

Note: Use of PGP for commercial purposes requires a license from Pretty Good Privacy, Inc. Please contact Pretty Good Privacy, Inc. for details and assistance in licensing PGP.

Use of PGP requires installation of PGP as well as generation and exchange of PGP public keys between the PMDF BSMTP systems which will be using PGP for authentication. This section documents step-by-step how to generate and exchange PGP keys. No attempt is here made to document PGP. Please refer to the documentation supplied with PGP for information on those subjects.

1. Acquire copies of PGP and install it on the PMDF systems. The following URLs might be helpful:

   ```
   <http://www.pgp.com/>
   <ftp://ftp.csn.net/mpj/getpgp.asc>
   <http://world.std.com/~franl/pgp/where-to-get-pgp.html>
   ```

2. After installing PGP, create an MTA key for the PMDF system. Note that the name for the key will be `bsmtp@bsin.`*`host0`* where *`host0`* is as in Section 23.1. The important element here is that the remote BSMTP channel will send the message to the address `bsmtp@bsin.`*`host0`*. The local PMDF system will receive that message and, via the `FORWARD` mapping table, route it to the incoming BSMTP channel, `bsin_gateway`, for the recipient `bsmtp@bsin.`*`host0`*. This recipient address is the user id of the decryption key which will be used.

   The PGP key rings need to be located somewhere; placing them in the directory `pmdf_root:[table.pgp]` is as good as place as any. The easiest way to set this up is as follows:

```
$ SET DEFAULT PMDF_ROOT:[TABLE]
$ CREATE/DIR/OWNER=[PMDF] [.PGP]
$ PGPPATH == "PMDF_ROOT:[TABLE.PGP]
$ PGP -kg
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 20:44 GMT
Pick your RSA key size:
    1)   512 bits- Low commercial grade, fast but less secure
    2)   768 bits- High commercial grade, medium speed, good security
    3)  1024 bits- "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits: 3
Generating an RSA key with a 1024-bit modulus.

You need a user ID for your public key.  The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example:  John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key: PMDF MTA key <bsmtp@bsin.host0>

You need a pass phrase to protect your RSA secret key.
Your pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

Enter pass phrase: secret
Enter same pass phrase again:  secret
Note that key generation is a lengthy process.

We need to generate 736 random bits.  This is done by measuring the
time intervals between your keystrokes.  Please enter some random text
on your keyboard until you hear the beep:
   0 * -Enough, thank you.
....**** ..............****
Key generation completed.
$ DIRECTORY/PROTECTION/SIZE=ALL [.PGP]
Directory PMDF_ROOT:[TABLE.PGP]

PUBRING.BAK;1              1/16       (RWED,RWED,,)
PUBRING.PGP;1              1/16       (RWED,RWED,,)
RANDSEED.BIN;1             1/16       (RWED,RWED,,)
SECRING.PGP;1              2/16       (RWED,RWED,,)

Total of 4 files, 5/64 blocks.
```

The final directory command verifies that the correct three PGP files have been created with appropriate rights.

3. You may want change the protections of the file `pubring.pgp` so that others can read the public key:

```
$ SET PROTECTION=(W:RE) [.PGP]PUBRING.PGP
```

4. Now you need to sign your public key. This prevents someone else from modifying the user id of the key.

```
$ PGP -ks "bsmtp@bsin.host0"
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 20:46 GMT

A secret key is required to make a signature.
You specified no user ID to select your secret key,
so the default user ID and key will be the most recently
added key on your secret keyring.

Looking for key for user 'bsmtp@bsin.host0':

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02
          Key fingerprint = 2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD

READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)? y

You need a pass phrase to unlock your RSA secret key.
Key for user ID "PMDF MTA key <bsmtp@bsin.host0>"

Enter pass phrase: secret
Pass phrase is good.  Just a moment....
Key signature certificate added.
```

5. Repeat Steps (1)—(4) on the other PMDF systems.

6. Next, you need to exchange public keys between the PMDF systems. On a given system, you may extract the public key as follows:

```
$ PGP -kxa "bsmtp@bsin.host0" extract
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 20:47 GMT

Extracting from key ring: '/pmdf/table/.pgp/pubring.pgp',
  userid "bsmtp@bsin.host0".

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02

Transport armor file: extract.asc
Key extracted to file 'extract.asc'.
```

The file `extract.asc` may then be transferred by FTP or e-mail to the other PMDF system. If you're exchanging keys with another server you control go on to the next step. However, if you're exchanging keys with a remote site, some care needs to be taken to make sure the public keys are properly certified.

7. The best way to exchange keys is to first exchange the key fingerprints via a reliable channel (*e.g.,* face-to-face in person, or perhaps over a trusted phone line). The fingerprint can be obtained with the following command:

```
$ PGP -kvc bsmtp@bsin.host0
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 23:08 GMT

Key ring: '/pmdf/table/.pgp/pubring.pgp', looking for user ID
  "bsmtp@bsin.host0".
Type bits/keyID    Date       User ID
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
          Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD
1 matching key found.
```

Then the public key itself can be extracted as described in Step (6) and sent through e-mail. Upon receipt, the fingerprint should be manually verified before certifying the key. After adding and certifying the key for the remote server, you may want to sign that key as well. If you sign the key, and extract it as described in Step (6) this can be used to tell other people you believe that key actually belongs to the MTA it claims to belong to. For more information, see the PGP documentation.

8. Add the key in `extract.asc` to the keyrings on the other PMDF systems. If you are unsure about how to answer the questions, see the *PGP User's Manual*.

```
$ PGP EXTRACT.ASC
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 21:09 GMT

File contains key(s).  Contents follow...
Key ring: 'extract.$00'
Type bits/keyID    Date       User ID
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
sig       BFFA43E9             PMDF MTA key <bsmtp@bsin.host0>
1 matching key found.

Do you want to add this keyfile to keyring '/pmdf/table/.pgp/pubring.pgp' (y/N)? y

Looking for new keys...
pub  1024/BFFA43E9 1997/04/02  PMDF MTA key <bsmtp@bsin.host0>

Checking signatures...
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
sig!      BFFA43E9 1997/04/02  PMDF MTA key <bsmtp@bsin.host0>

Keyfile contains:
   1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)? y
```

```
Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02
Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD
This key/userID association is not certified.
  Questionable certification from:
  PMDF MTA key <bsmtp@bsin.host0>

Do you want to certify this key yourself (y/N)? y

Looking for key for user 'PMDF MTA key <bsmtp@bsin.host0>':

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02
          Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD

 READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)? y

You need a pass phrase to unlock your RSA secret key.
Key for user ID "PMDF MTA key <bsmtp@bsin.host1>"

Enter pass phrase: another-secret
Pass phrase is good.  Just a moment....
Key signature certificate added.

Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on available
evidence.  If you think it does, then based on your estimate of
that person's integrity and competence in key management, answer
the following question:

Would you trust "PMDF MTA key <bsmtp@bsin.host0>"
to act as an introducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ? 2
```

9. Repeat Steps (6)—(8) in the other direction.

10. You may check which keys are on your keyring with the following command:

```
$ PGP -kv
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 23:23 GMT

Key ring: '/pmdf/table/.pgp/pubring.pgp'
Type bits/keyID    Date       User ID
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
pub  1024/6405957D 1997/03/17 PMDF MTA key <bsmtp@bsin.host0>
2 matching keys found.
```

Once you have exchanged the keys, you should then be able to send digitally signed BSMTP parcels.

23.2.3 Examples on UNIX

The following sections provide examples of using BSMTP channels on UNIX.

23.2.3.1 Configuring the BSMTP Channels to Compress Their Payloads on UNIX

Using PMDF's general purpose, on-the-fly conversion facilities, BSMTP parcels can be compressed on the sending system and then uncompressed on the receiving system. This allows for faster transmission of the parcels through the network.

In the CHARSET-CONVERSION mapping table on each PMDF system, a simple entry enabling conversions for the BSMTP channels must be made:

```
CHARSET-CONVERSION

  in-chan=bsout_*;out-chan=*;convert      yes
  in-chan=*;out-chan=bsin_*;convert       yes
```

In the PMDF conversions file on each system, conversion entries are added which call out to the site-supplied shell script, compress.sh:

```
in-chan=bsout_*; part-number=1; in-type=*; in-subtype=*;
  service-command="/pmdf/bin/compress.sh compress $INPUT_FILE $OUTPUT_FILE"

out-chan=bsin_*; part-number=1; in-type=application;
  in-subtype=compressed-bsmtp;
  service-command="/pmdf/bin/compress.sh decompress $INPUT_FILE $OUTPUT_FILE"
```

The compress.sh shell script is shown in Figure 23–4. It assumes that the gzip and gunzip utilities are installed in /usr/local/bin/.

23.2.3.2 Configuring the BSMTP Channels to Provide Authentication Services on UNIX

Using PMDF's general purpose, on-the-fly conversion facilities, authentication and integrity services may be tied in to the BSMTP channels. This is done through the CHARSET-CONVERSION mapping table, the PMDF conversions file, and a site-supplied shell script to digitally sign payloads and verify the signature and integrity of the data upon receipt.

In the CHARSET-CONVERSION mapping table on each PMDF system, a simple entry enabling conversions for the BSMTP channels must be made:

```
CHARSET-CONVERSION

  in-chan=bsout_*;out-chan=*;convert      yes
  in-chan=*;out-chan=bsin_*;convert       yes
```

Figure 23–4 `compress.sh`**: Compress and decompress BSMTP payloads**

```
#!/sbin/sh

# compress operation in-file out-file [addr-file]

# where

#   operation   == "compress" | "decompress"
#   input-file  == path of file to sign or verify
#   output-file == output file to produce
#   addr-file   == file of envelope recipient addresses

if [ $# -lt 3 ]; then exit 1; fi

case $1
in
  compress )
        /usr/local/bin/gzip < $2 > $3.tmp
        /pmdf/bin/pmdf encode -nofilename -encoding=base64 -type=application \
                -subtype=compressed-bsmtp $3.tmp $3.tmp2
        rm -f $3.tmp $3.tmp2
        ;;

  decompress )
        /pmdf/bin/pmdf decode $2 $3.tmp
        /usr/local/bin/gunzip < $3.tmp > $3
        rm -f $3.tmp
        ;;

  * )
        exit 1
        ;;
esac
exit 0
```

In the PMDF conversions file file on each system, there must be conversion entries to invoke the site-supplied shell scripts:

```
in-chan=bsout_*; part-number=1; in-type=*; in-subtype=*;
  service-command="/pmdf/bin/pgp_sign.sh $INPUT_FILE $OUTPUT_FILE"

out-chan=bsin_*; part-number=1; in-type=multipart; in-subtype=signed;
  service-command="/pmdf/bin/pgp_verify.sh $INPUT_FILE $OUTPUT_FILE"
```

These two scripts are shown in Figures 23–5 and 23–6. They assume that the `pgp` utility is installed in `/usr/local/bin/` and that `awk` is installed in `/usr/bin/`. Note that the `pgp_sign.sh` script requires the pass phrase for the PMDF MTA's private PGP key in order to generate signatures. Edit the script to reflect the correct pass phrase and be sure to protect the file from other users:

```
% chown pmdf:bin /pmdf/bin/pgp_sign.sh
% chmod 0700 /pmdf/bin/pgp_sign.sh
```

Figure 23–5 `pgp_sign.sh`**: Digitally sign BSMTP payloads**

```
#!/sbin/sh

# pgp_sign.sh input-file output-file [addr-file]

# where

#   input-file  == path of file to sign or verify
#   output-file == output file to produce
#   addr-file   == file of envelope recipient addresses

# Check that we have at least three command line parameters
if [ $# -lt 2 ]; then exit 1; fi

# Change these to match your site
PGPUSER="PMDF MTA key"
PGPPATH=/pmdf/table/pgp
PGPPASS="Percy eats pealed banannas"

# Generate the digital signature
/usr/local/bin/pgp -sab $1 -u $PGPUSER -z $PGPPASS -o $2 +batchmode

# Make some temporary files used to MIME-ify the results
BOUNDARY=`/pmdf/bin/unique_id`
echo 'Content-type: multipart/signed; boundary="'$BOUNDARY'"; '\
'micalg=pgp-md5; protocol=application/pgp-signature

--'$BOUNDARY > $2.top
echo '--'$BOUNDARY'
Content-type: application/pgp-signature
' > $2.mid
echo --$BOUNDARY-- > $2.bot

# Make a multipart/signed message part
cat $2.top $1 $2.mid $2.asc $2.bot > $2

# Now clean up
rm -f $2.top $2.mid $2.asc $2.bot

# And exit
exit 0
```

23.2.3.2.1 Using PGP with PMDF on UNIX

Note: Use of PGP for commercial purposes requires a license from Pretty Good Privacy, Inc. Please contact Pretty Good Privacy, Inc. for details and assistance in licensing PGP.

Use of PGP requires installation of PGP as well as generation and exchange of PGP public keys between the PMDF BSMTP systems which will be using PGP for authentication. This section documents step-by-step how to generate and exchange PGP keys. No attempt is here made to document PGP. Please refer to the documentation supplied with PGP for information on those subjects.

1. Acquire copies of PGP and install it on the PMDF systems. The following URLs might be helpful:

Figure 23–6 `pgp_verify.sh`: **Verify the integrity of a digitally signed BSMTP payload**

```
#!/sbin/sh

# pgp_verify.sh input-file output-file [addr-file]

# where

#   input-file  == path of file to sign or verify
#   output-file == output file to produce
#   addr-file   == file of envelope recipient addresses

# Check that we have at least three command line parameters
if [ $# -lt 2 ]; then exit 1; fi

# Change this to match your site
PGPPATH=/pmdf/table/pgp

# Use awk to split the multipart/signed part into
# two files: the signed data and the digital signature
/usr/bin/awk '
BEGIN { state = 0 }
{
        if (state == 0) {
                if (substr ($0, 0, 2) == "--") {
                        boundary = $0
                        state = 1
                }
        } else if (state == 1) {
                if ($0 != boundary) {
                        print $0 > OUT_DATA
                } else {
                        state = 2
                }
        } else if (state == 2) {
                if (NF == 0) state = 3
        } else if (state == 3) {
                print $0 > OUT_SIGN
        }
}' OUT_DATA=$2.data OUT_SIGN=$2.sign $1

# Verify the digital signature
/usr/local/bin/pgp $2.sign $2.data +batchmode > $2.check

# Build a X-Content-MIC-check: header line
MICINFO=`grep -h ' signature from user ' $2.check`
if [ -n "$MICINFO" ]
then
        echo 'X-Content-MIC-check: '$MICINFO > $2.mic
else
        echo 'X-Content-MIC-check: Bad signature' > $2.mic
fi
cat $2.mic $2.data > $2

# Clean up
rm -f $2.sign $2.data $2.check $2.mic

# And exit
exit 0
```

```
<http://www.pgp.com/>
<ftp://ftp.csn.net/mpj/getpgp.asc>
<http://world.std.com/~franl/pgp/where-to-get-pgp.html>
```

2. After installing PGP, create an MTA key for the PMDF system. Note that the name for the key will be

   PMDF MTA key bsmtp@bsin.*host0*

   bsmtp@bsin.*host0* where *host0* is as in Section 23.1. The important element here is that the remote BSMTP channel will send the message to the address bsmtp@bsin.*host0*. The local PMDF system will receive that message and, via the FORWARD mapping table, route it to the incoming BSMTP channel, bsin_gateway, for the recipient bsmtp@bsin.*host0*. This recipient address is the user id of the decryption key which will be used.

   The PGP key rings need to be located somewhere; placing them in the directory /pmdf/table/pgp/ is as good as place as any. The easiest way to set this up is as follows:

```
% su pmdf
% cd /pmdf/table
% mkdir pgp
% setenv PGPPATH /pmdf/table/pgp
% pgp -kg
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 20:44 GMT
Pick your RSA key size:
    1)   512 bits- Low commercial grade, fast but less secure
    2)   768 bits- High commercial grade, medium speed, good security
    3)  1024 bits- "Military" grade, slow, highest security
Choose 1, 2, or 3, or enter desired number of bits: 3
Generating an RSA key with a 1024-bit modulus.

You need a user ID for your public key.  The desired form for this
user ID is your name, followed by your E-mail address enclosed in
<angle brackets>, if you have an E-mail address.
For example:  John Q. Smith <12345.6789@compuserve.com>
Enter a user ID for your public key: PMDF MTA key <bsmtp@bsin.host0>

You need a pass phrase to protect your RSA secret key.
Your pass phrase can be any sentence or phrase and may have many
words, spaces, punctuation, or any other printable characters.

Enter pass phrase: secret
Enter same pass phrase again:  secret
Note that key generation is a lengthy process.
```

```
We need to generate 736 random bits.  This is done by measuring the
time intervals between your keystrokes.  Please enter some random text
on your keyboard until you hear the beep:
   0 * -Enough, thank you.
....**** ..............****
Key generation completed.
% ls -l pgp
total 6
-rw-------   1 pmdf     30           197 Apr  2 12:52 pubring.pgp
-rw-------   1 pmdf     30           408 Apr  2 12:52 randseed.bin
-rw-------   1 pmdf     30           530 Apr  2 12:52 secring.pgp
```

The final `ls` command verifies that the correct three PGP files have been created with appropriate rights.

3. You may want change the permissions of the file `pubring.pgp` so that others can read the public key:

```
% chmod 644 pgp/pubring.pgp
% ls -l pgp
total 6
-rw-r--r--   1 pmdf     30           197 Apr  2 12:52 pubring.pgp
-rw-------   1 pmdf     30           408 Apr  2 12:52 randseed.bin
-rw-------   1 pmdf     30           530 Apr  2 12:52 secring.pgp
```

4. Now you need to sign your public key. This prevents someone else from modifying the user id of the key.

```
% pgp -ks bsmtp@bsin.host0
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 20:46 GMT

A secret key is required to make a signature.
You specified no user ID to select your secret key,
so the default user ID and key will be the most recently
added key on your secret keyring.

Looking for key for user 'bsmtp@bsin.host0':

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02
          Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD

READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)? y

You need a pass phrase to unlock your RSA secret key.
Key for user ID "PMDF MTA key <bsmtp@bsin.host0>"

Enter pass phrase: secret
Pass phrase is good.  Just a moment....
Key signature certificate added.
```

5. Repeat Steps (1)—(4) on the other PMDF systems.

6. Next, you need to exchange public keys between the PMDF systems. On a given system, you may extract the public key as follows:

```
% pgp -kxa bsmtp@bsin.host0 extract
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 20:47 GMT

Extracting from key ring: '/pmdf/table/.pgp/pubring.pgp',
  userid "bsmtp@bsin.host0".

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02

Transport armor file: extract.asc
Key extracted to file 'extract.asc'.
```

The file `extract.asc` may then be transferred by FTP or e-mail to the other PMDF system. If you're exchanging keys with another server you control go on to the next step. However, if you're exchanging keys with a remote site, some care needs to be taken to make sure the public keys are properly certified.

7. The best way to exchange keys is to first exchange the key fingerprints via a reliable channel (*e.g.,* face-to-face in person, or perhaps over a trusted phone line). The fingerprint can be obtained with the following command:

```
% pgp -kvc bsmtp@bsin.host0
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 23:08 GMT

Key ring: '/pmdf/table/.pgp/pubring.pgp', looking for user ID
  "bsmtp@bsin.host0".
Type bits/keyID    Date       User ID
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
          Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD
1 matching key found.
```

Then the public key itself can be extracted as described in Step (6) and sent through e-mail. Upon receipt, the fingerprint should be manually verified before certifying the key. After adding and certifying the key for the remote server, you may want to sign that key as well. If you sign the key, and extract it as described in Step (6) this can be used to tell other people you believe that key actually belongs to the MTA it claims to belong to. For more information, see the PGP documentation.

8. Add the key in `extract.asc` to the keyrings on the other PMDF systems. If you are unsure about how to answer the questions, see the *PGP User's Manual*.

```
% pgp extract.asc
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 21:09 GMT

File contains key(s).  Contents follow...
Key ring: 'extract.$00'
Type bits/keyID    Date       User ID
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
sig       BFFA43E9            PMDF MTA key <bsmtp@bsin.host0>
1 matching key found.

Do you want to add this keyfile to keyring '/pmdf/table/.pgp/pubring.pgp' (y/N)? y

Looking for new keys...
pub  1024/BFFA43E9 1997/04/02  PMDF MTA key <bsmtp@bsin.host0>

Checking signatures...
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
sig!      BFFA43E9 1997/04/02  PMDF MTA key <bsmtp@bsin.host0>

Keyfile contains:
   1 new key(s)

One or more of the new keys are not fully certified.
Do you want to certify any of these keys yourself (y/N)? y

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02
Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD
This key/userID association is not certified.
  Questionable certification from:
  PMDF MTA key <bsmtp@bsin.host0>

Do you want to certify this key yourself (y/N)? y

Looking for key for user 'PMDF MTA key <bsmtp@bsin.host0>':

Key for user ID: PMDF MTA key <bsmtp@bsin.host0>
1024-bit key, Key ID BFFA43E9, created 1997/04/02
          Key fingerprint =  2F 5C A1 0A 35 25 E1 23  ED AF 23 11 00 37 5A CD

 READ CAREFULLY:  Based on your own direct first-hand knowledge, are
you absolutely certain that you are prepared to solemnly certify that
the above public key actually belongs to the user specified by the
above user ID (y/N)? y

You need a pass phrase to unlock your RSA secret key.
Key for user ID "PMDF MTA key <bsmtp@bsin.host1>"

Enter pass phrase: another-secret
Pass phrase is good.  Just a moment....
Key signature certificate added.
```

```
Make a determination in your own mind whether this key actually
belongs to the person whom you think it belongs to, based on available
evidence.  If you think it does, then based on your estimate of
that person's integrity and competence in key management, answer
the following question:

Would you trust "PMDF MTA key <bsmtp@bsin.host0>"
to act as an introducer and certify other people's public keys to you?
(1=I don't know. 2=No. 3=Usually. 4=Yes, always.) ? 2
```

9. Repeat Steps (6)—(8) in the other direction.

10. You may check which keys are on your keyring with the following command:

```
% pgp -kv
Pretty Good Privacy(tm) 2.6.2 - Public-key encryption for the masses.
(c) 1990-1994 Philip Zimmermann, Phil's Pretty Good Software. 11 Oct 94
Uses the RSAREF(tm) Toolkit, which is copyright RSA Data Security, Inc.
Distributed by the Massachusetts Institute of Technology.
Export of this software may be restricted by the U.S. government.
Current time: 1997/04/02 23:23 GMT

Key ring: '/pmdf/table/.pgp/pubring.pgp'
Type bits/keyID    Date       User ID
pub  1024/BFFA43E9 1997/04/02 PMDF MTA key <bsmtp@bsin.host0>
pub  1024/6405957D 1997/03/17 PMDF MTA key <bsmtp@bsin.host0>
2 matching keys found.
```

Once you have exchanged the keys, you should then be able to send digitally signed BSMTP parcels.

24 PhoneNet Channels (OpenVMS and UNIX)

PhoneNet is an asynchronous terminal line-based transport designed for use with PMDF, MMDF, and CSNET. When it operates in master mode, PhoneNet is capable of "dialing out" over telephone lines to establish a connection with a PhoneNet slave on another machine. When it operates in slave mode, PhoneNet responds to a remote master's requests for message transfers.

PhoneNet is divided into two separate protocols. The lower "link-level" protocol, called the "dial protocol", provides a basic byte-stream environment. It is responsible for connection acquisition and management, character set translation and flow control. Dial is only used on asynchronous terminal lines.

The higher-level protocol, called the "phone protocol", is responsible for the semantics of mail transfer. It must establish the context and then pass individual messages. The overall structure of PhoneNet and the interaction of these two protocols is shown in Figure 24–1.

The following sections describe many, but not all, aspects of PhoneNet. In particular, the various input and output files used by PhoneNet channels are described in detail. However, the internal format of the dial and phone protocols are not described here. Refer to the files `phone.ovr` and `dialproto.col` in PMDF's documentation directory, (`PMDF_DOC:` on OpenVMS, or `/pmdf/doc/` on UNIX, or usually `C:\pmdf\doc\` on NT), for information on, respectively, the phone and dial protocols.

Four steps are required to set up a PhoneNet channel: (1) Add the channel to your PMDF configuration, (2) Create an option file for the channel, (3) Create a `phone_list.dat` file listing your serial devices, and (4) Create a script file that provides the necessary modem commands and login sequences. Each of these steps is described in detail in the following sections.

PhoneNet channels are not generated by the automatic configuration generator.

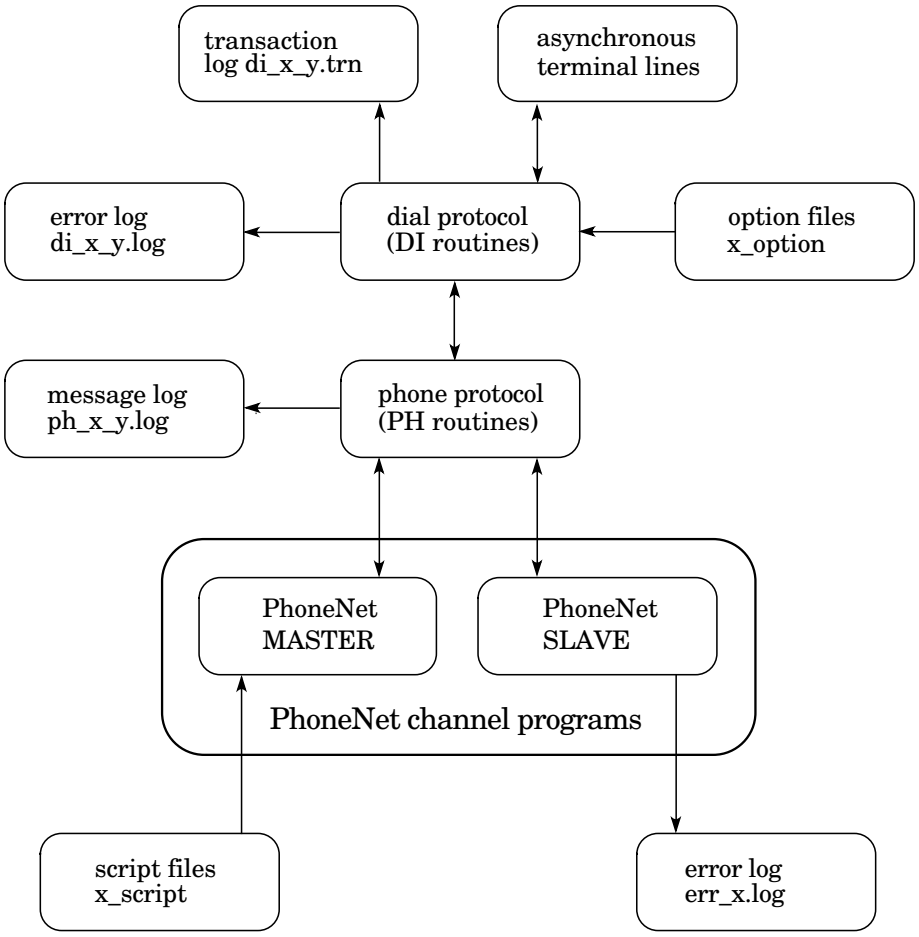
24.1 Adding PhoneNet channels to the configuration file

The first step is to add the PhoneNet channel to the PMDF configuration file. PhoneNet channel names must begin with "p_", for example p_smc.

Note: For compatability with previous releases, PMDF on OpenVMS will allow PhoneNet channel names without the "p_" prefix. Such names are not recommended.

Since PhoneNet channels are point-to-point, you must create a separate channel, with its own option and script files, for each link to a remote system. A typical channel definition looks like this:

Figure 24–1 The Structure of PhoneNet




```
p_smc
engvax.smc.com
```

The rewrite rules in the configuration files should now be edited to refer to the new channel as appropriate. Continuing our example, a likely rewrite rule for the channel definitions shown above:

```
engvax.smc.com       $U@engvax.smc.com
```

24.2 PhoneNet option files

Option files are used to set several run-time PhoneNet options on a per-channel basis. Option files are stored in the PMDF table directory (`PMDF_TABLE:` on OpenVMS, or `/pmdf/table/` on UNIX, or typically `C:\pmdf\table\` on NT) and have names of the form *x*`_option`, where "*x*" is the name of the PhoneNet channel to which the option file applies. Every PhoneNet channel must have an option file.

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

```
option=value
```

where *value* can be either a string or an integer, depending on the option's requirements. If the option accepts an integer value, *value*, a base can be specified using notation of the form *b*%*v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

Comments are allowed. Any line that begins with an exclamation point is considered to be a comment and is ignored. Blank lines are also ignored in any option file.

The available options are:

BACKOFF (integer >= 0)
When BACKOFF specifies an integer greater than zero, backoff retries will be attempted for messages which could not be delivered immediately. See Section 24.6 for details. By default, BACKOFF=0.

BAUDRATE (integer)
This option specifies the baud rate at which to dial-out using the MASTER program. Possible values are: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19200, and 38400. Not all hardware interfaces support all these baud rates. In particular, Local Area Terminal Server (LAT) lines can ignore the baud rate setting on the host side; the line speed might need to be set from the terminal server itself (which PMDF cannot access) to have any effect.[1]

If no BAUDRATE option is specified in the option file the MASTER program does not change the baud rate of the terminal. On OpenVMS, it is also possible to set the baud rate using a SET TERMINAL command in the `all_master.com` command file (see Section 24.3.2).

CHANNEL (string)
The CHANNEL option defines the name of the channel with which the option file is associated. This name is used to generate Received: header lines in message headers. The CHANNEL option *must* be specified for any channel other than the default PhoneNet channel (channel p); if it is not, the channel will not work properly.

LOGGING (0 or 1)
LOGGING controls whether or not the channel maintains a PhoneNet message log file `ph_`*x*`_`*y*`.log`. A value of 1, the default, specifies that a log is to be kept. A value of 0 disables use of the PhoneNet log file.

TRANSCRIBE (0 or 1)
TRANSCRIBE controls whether or not the channel maintains a Dial protocol transaction log file `di_`*x*`_`*y*`.log`. A value of 1, the default, specifies that a log is to be kept. A value of 0 disables use of the transaction log file.

[1] Some terminal servers allow the host to alter a port's speed provided the port has the "remote modification" characteristic enabled.

DATAWAIT (integer)
This defines the number of seconds to wait for a data packet from the remote host. Normally the default value of 180 suffices. A higher value can be necessary if the remote host runs very slowly.

XMITWAIT (integer)
This is the number of seconds to wait for a packet to be transmitted. The default value of 60 should work for almost all applications.

QACKWAIT (integer)
This is the number of seconds to wait for a response to a QUIT packet before retransmission. The default value of 30 should work for almost all applications.

EACKWAIT (integer)
This is the number of seconds to wait for a response to an ESCAPE packet before resending. The default value of 30 should work for almost all applications.

DACKWAIT (integer)
This is the number of seconds to wait for data acknowledgement packets before retransmission. The default value of 30 should work for almost all applications.

ESCAPEWAIT (integer)
This is the number of seconds to wait for an ESCAPE packet during protocol startup. The default value of 60 should work for almost all applications.

XPATHWAIT (integer)
This is the number of seconds to wait for an XPATH packet during protocol startup. The default value of 60 is usually sufficient. However, since this is the first packet exchanged in PhoneNet, it can be necessary to increase this parameter on systems which process lots of files (tallying files to be transmitted delays the sending of the first packet) or which take a long time to start up the PhoneNet programs.

RPATHWAIT (integer)
This is the number of seconds to wait for RPATH packets. The default value of 60 should work for almost all applications.

NBUFFWAIT (integer)
This is the number of seconds to wait for NBUFF packets. The default value of 60 should work for almost all applications.

XPAWAIT (integer)
This is the number of seconds to wait for XPATH packet acknowledgements. The default value of 20 should work for almost all applications.

RPAWAIT (integer)
This is the number of seconds to wait for RPATH packet acknowledgements. The default value of 20 should work for almost all applications.

NBAWAIT (integer)
This is the number of seconds to wait for NBUFF packet acknowledgements. The default value of 20 should work for almost all applications.

EIGHTBIT (0 or 1)

The EIGHTBIT option controls whether eight-bit characters are enabled by MASTER. On OpenVMS, this corresponds to the eightbit setting of the OpenVMS terminal driver. If EIGHTBIT has a value of 0 the eighth bit is masked out on output characters and ignored on input characters. If EIGHTBIT is 1 the eighth bit is preserved on output and interpreted on input. Any other value or the omission of the EIGHTBIT option will tell MASTER not to change the interpretation of the EIGHTBIT flag from its current setting.

On OpenVMS, it is also possible to set this option using a SET TERMINAL command in the `all_master.com` command file as described in Section 24.3.2.

PhoneNet normally quotes eight-bit characters so the setting of this characteristic is not relevant insofar as the PhoneNet protocol is concerned. However, it can be necessary to enable or disable the EIGHTBIT option in order for the MASTER login script to work properly.

EXPECTWINDOW (0 or 1)

This option tells the local slave program whether to expect a "set window size" packet from the remote master. The default is 0 which means that the packet *must not* be received. A setting of 1 means that the packet *must* be received. If EXPECTWINDOW is set to 1, the window size that is received in the "set window size" packet overrides the setting of the WINDOW option.

FLUSHRATE (integer)

The FLUSHRATE parameter controls how often transaction log file buffers are emptied and the files brought up to date. PMDF periodically issues a $FLUSH on the transaction log. This operation, coupled with the fact that the file is shareable, makes it possible to examine the file while MASTER or SLAVE is still running. (On OpenVMS, the standard DCL TYPE command can be used to examine the file if desired.) The FLUSHRATE value is expressed as the number of lines between flush operations. The minimum value of 1 means that the file will be updated as each line of text is written to it. Low values can cause performance degradation due to the expense of issuing so many $FLUSH calls. The default value is 32.

HOSTSYNC (0 or 1)

This option tells PMDF whether or not to use flow control (CTRL/S and CTRL/Q) to regulate incoming data flow. If HOSTSYNC's value is 1 PMDF will instruct the system to send a CTRL/S when the terminal input buffer is nearly full and a CTRL/Q when the buffer empties. A HOSTSYNC value of 0 tells PMDF to disable this form of flow control. Any other value or not specifying the HOSTSYNC will leave this setting unchanged from the default already set on the terminal.

The HOSTSYNC setting on one side of a connection should usually match the TTSYNC setting on the other, and vice versa.

On OpenVMS, it is also possible to set this option using a SET TERMINAL command in the `all_master.com` command file as described in Section 24.3.2. Setting this option in this command file will not affect its use by SLAVE, however.

NOISE (list of integers)

This defines the set of characters to ignore when reading data from the remote host. The decimal ASCII values of the characters to ignore are listed separated by commas. The default is 13 (CR).

PARITY (NONE, ODD, EVEN, MARK, SPACE)
This controls the use of parity on the line. Possible values are NONE, ODD, EVEN, MARK, and SPACE. PMDF will assert the specified parity on outgoing characters. Any setting other than NONE will cause the uppermost (eighth) bit to be stripped from all incoming characters. A parity setting of NONE passes all characters as-is. The default is NONE.

SENDWINDOW (0 or 1)
This tells the local master program whether to send a "set window size" packet to the remote slave. The default is 0 (no). A setting of 1 means yes. Some older link-level protocol implementations do not accept this "set window size" packet. The actual window size is determined by the WINDOW option setting described above. The default window size on relay.cs.net is 1 so set WINDOW=2 and SENDWINDOW=1 to override this default.

TERMINATOR (integer)
This defines the packet-terminator for packets sent to the remote host. The decimal ASCII values of the desired terminators should be listed without quotes and separated by commas. The default is "13,10" (CR,LF).

TTSYNC (0 or 1)
This option tells PMDF whether or not to interpret flow control (CTRL/S and CTRL/Q) sent by the remote system. If TTSYNC's value is 1 PMDF will instruct the system to stop sending characters when a CTRL/S is received and to resume sending when a CTRL/Q is received. A TTSYNC value of 0 tells PMDF to disable this form of flow control. Any other value or not specifying the TTSYNC will leave this setting unchanged from the default already set on the terminal.

On OpenVMS, it is also possible to set this option using a SET TERMINAL command in the `all_master.com` command file as described in Section 24.3.2. Setting this option in this command file will not affect its use by SLAVE, however.

The TTSYNC setting on one side of a connection should usually match the HOSTSYNC setting on the other, and vice versa.

WINDOW (1 or 2)
This defines the window size for the link-level PhoneNet protocol. The default is 1 (single buffering). A setting of 2 means double buffering. Double buffering can be used only over full-duplex lines and typically results in 25% to 60% throughput improvements, depending on the type of communication line being used. The local and remote hosts must agree on the window size. (See SENDWINDOW and EXPECTWINDOW above.)

24.2.1 An example option file

Continuing our example channel from above, the following is an example option file demonstrating typical options that are specified. Only the CHANNEL option is actually required.

```
CHANNEL=p_smc
TERMINATOR=10
DATAWAIT=180
WINDOW=2
SENDWINDOW=1
EXPECTWINDOW=1
```

24.3 Defining serial devices

PhoneNet and Pager channels use the file `phone_list.dat` in the PMDF table directory to relate transport methods to certain channels. Each line in this file is referred to as a "method" and has the format

channel device [*script* [*options*]]

where *channel* is the name of a channel, *device* is a possible device that the channel can use, *script* is the name used to find the script file, and *options* are a set of options passed to a user-written command file. The `options` field is ignored on UNIX platforms.

device is usually, but not always, the name of an asynchronous terminal device. On OpenVMS, it can be a logical name which translates to one or more devices. If *options* are omitted, an empty set of options is used. If *script* and *options* are both omitted an empty set of options are used and *script* is set to *device*.

The usual form of PMDF comment lines are all allowed in the file. Channel names are not case sensitive. The *device*, *options*, and *script* are not case sensitive on OpenVMS, but are case sensitive on UNIX platforms.

Example 24–1 depicts a typical `phone_list.dat` file for OpenVMS. Example 24–2 depicts a typical `phone_list.dat` file for UNIX systems.

Example 24–1 OpenVMS Sample PhoneNet `phone_list.dat` file

```
! phone_list.dat - Allow the use of either modem
! for Pager channel.
! There are two numbers to try on P_SMC channel,
! so try it with two different scripts.
!
pager_pactel   SLOW_MODEM   pager_slow
pager_pactel   AUTO_DIALER  pager_fast
!
p_smc     AUTO_DIALER   dial_one
p_smc     AUTO_DIALER   dial_two
```

Example 24–2 UNIX Sample PhoneNet `phone_list.dat` file

```
! phone_list.dat - Allow the use of either slow or fast modem
! for Pager channel.
```

Example 24–2 Cont'd on next page

Example 24–2 (Cont.) UNIX Sample PhoneNet `phone_list.dat` file

```
! There are two numbers to try on P_SMC channel,
! so try it with two different scripts but only on fast modem.
!
pager_pactel   /dev/ttyd0  pager_slow
pager_pactel   /dev/ttyd1  pager_fast
!
p_smc     /dev/ttyd1   dial_one
p_smc     /dev/ttyd1   dial_two
```

A channel can have one or more entries associated with it. In the two examples, two entries are present for each channel. For the pager_pactel channel we want to use two different devices. For the p_smc channel we only want to use one device, but we want to try with two different scripts.

On OpenVMS, multiple devices such as a modem pool, can be used by specifying a logical name for *device* which translates to the list of devices to use. For example, the logical AUTO_DIALER used in Example 24–1 defined as

```
$ DEFINE/SYSTEM AUTO_DIALER LTA100:,TXA6:,LTA2:,LTA5:
```

is a search list specifying the four devices LTA100:, TXA6:, LTA2:, and LTA5:.

On UNIX and NT platforms, multiple devices need to be handled with multiple entries in the `phone_list.dat` file for each channel, one entry per channel per device.

24.3.1 Script files

The final piece of information needed for PhoneNet and Pager channels is the connection script to use for each particular *channel*, *device*, and *script* combination. The *script* field of an entry identifies a file containing the PhoneNet script. The script is located in a subdirectory under the PMDF table directory. The subdirectory has the same name as the channel. On OpenVMS systems, the script file would be named as `pmdf_root:[table.`*channel*`]`*script*`_script.`; on UNIX platforms, the script file would be named as `/pmdf/table/`*channel*`/`*script*`_script`; on NT systems, the script file typically would be named as `C:\pmdf\table\`*channel* `\`*script*`_script`.

Different or common files can be used for each *channel*, *device*, *script* combination. Each connection can potentially require a completely different script, or it can share a script with many other entries, depending on the application. Note the use of an individual subdirectory for each channel.

The format of script files is described in Section 24.5.

Note: It is customary to protect script files from world or group access: script files generally contain the information required to login on remote system running PhoneNet. If that remote system runs PMDF, then the remote account will be a restricted account. But, nonetheless, it is best to protect this information.

24.3.2 The all_master.com file (OpenVMS)

Note: On UNIX platforms there is no equivalent of the `all_master.com` procedure. The PhoneNet and Pager channels set all terminal line characteristics from the channel options file.

On OpenVMS platforms, once a method is found, `master.com` executes the procedure `all_master.com` to set up and condition the terminal line. This file is used in lieu of the usual *x*`_master.com` command file that `master.com` invokes. `all_master.com` is invoked with *device* as its first parameter and *options* as its second parameter. These two parameters are drawn from the `phone_list.dat` file which is described in Section 24.3 above. A typical `all_master.com` file is shown in Example 24–3. You should supply what is appropriate for your terminals in your own `all_master.com`.

Example 24–3 A Sample PhoneNet `all_master.com` file

```
$ ! all_master.com - A sample all_master.com file
$ !
$ IF P1 .NES. "" THEN GOTO USE_'P1'
$ !
$ USE_AUTO_DIALER:
$       ALLOCATE AUTO_DIALER
$       SET TERM/SPEED=1200 AUTO_DIALER
$       DEFINE TT AUTO_DIALER
$       EXIT
$ !
$ USE_MUX_LINE:
$       ALLOCATE MUX_LINE
$       DEFINE TT MUX_LINE
$       EXIT
```

The conditions and restrictions that apply to *x*`_master.com` files also apply to `all_master.com`. TT can be defined to point at a multi-equivalence logical name (*i.e.,* a search list) as in the case of *x*`_master.com`. If a search list is used the ALLOCATE commands should be removed.

24.3.3 Handling failures

If a particular method from the `phone_list.dat` file fails to provide a PhoneNet connection for whatever reason (terminal unavailable, script timeout, no answer, bad transmission line, no free ports, *etc.*), then PMDF resumes the search through `phone_list.dat`, trying each method that it finds for the current channel until one succeeds or the file is exhausted. Entries can be repeated if multiple connection attempts using the same method are desired.

On OpenVMS, if a failure occurs within a section of script marked as an initialization section, then before trying the next method, any additional devices specified by *device* will first be tried. Note that in order for *device* to specify more than one device, it must be a logical search list as described in Section 24.3. Initialization sections of a script file are marked with the "init" script command.

24.4 Log files

Four log files are produced by PhoneNet as it operates. These log files are useful both for keeping track of PhoneNet activity and for diagnosing problems with PhoneNet as they occur.

All PhoneNet log files are kept in the PMDF log directory, (*i.e.*, in `PMDF_LOG:` on OpenVMS, or in `/pmdf/log/` on UNIX, or in `C:\pmdf\log\` on NT). A separate copy of each log file is maintained for each PhoneNet channel and for each master and slave. A typical log file name is `ph_`*x*`_`*y*`.log`, where *x* is the name of the channel with which the log file is associated and *y* is either `master` for an outbound call or `slave` for an inbound call.

The log files are:

Dial protocol transaction log (`di_`*x*`_`*y*`.trn`)
This file contains a complete transcript of the most recent PhoneNet session. This log shows the low-level protocol and is generally useful only for purposes of pinpointing where a failure is occurring. A new version of this file is created for each session. The TRANSCRIBE channel option can be used to disable creation and maintenance of this log file if desired.

Dial protocol error log (`di_`*x*`_`*y*`.log`)
This file contains a list of all the transmission errors detected by the dial protocol (DI routines). It should be examined and purged periodically by the system manager. This file is appended to rather than recreated for each session. This file will be created automatically if it does not exist.

Message log (`ph_`*x*`_`*y*`.log`)
This file contains a list of all the messages that have been transmitted or received. It is useful for keeping track of the number and volume of messages handled by PhoneNet. This file is appended to rather than recreated for each session. This file will be created automatically if it does not exist. The LOGGING channel option can be used to disable creation and maintenance of this log file if desired.

With the addition of a general logging facility activated by the `logging` channel keyword in PMDF V3.0, this log file has become redundant. It will be removed in some future release of PMDF. Sites that use the information logged in this file are advised to revise their software to use the information logged in the general PMDF log file instead.

SLAVE error log (`err_`*x*`.log`)
This file contains any reports of errors detected during the operation of the SLAVE program. Its contents are generally not useful. A new version of this file is created each time SLAVE runs.

Additional log files are created by the jobs that run MASTER either in response to message queueing or for polling purposes. These files are described in detail in the section on PMDF command files below.

24.5 Script files

Script files contain a series of commands which tell MASTER what to send to the terminal in order to establish a connection with a remote PhoneNet slave. These files are all located in one of the channel-specific subdirectories of the PMDF table directory, as explained in Section 24.3.1.

Script file commands are provided to send strings to a serial device, to wait until a specified string is received and to terminate script processing and start up the dial protocol.

The available commands are described below. The *quotedstring* appearing in the descriptions is any sequence of characters enclosed in double quotes.

xmit *quotedstring*
Send *quotedstring* to the terminal line. The sequence `\x` in the string causes a one-second transmission delay at the point where it appears in the string.

recv *quotedstring timeout*
Wait until *quotedstring* is received before proceeding. Any characters leading up to *quotedstring* are ignored. Script processing is terminated if the *quotedstring* is not received within *timeout* seconds and timeout termination processing is enabled (the default).

toff
Disable timeout termination processing on all subsequent recv commands. Any timeouts that occur are ignored.

ton
Enable timeout termination processing on all subsequent recv commands. Any timeouts that occur will cause program termination. This is the default unless a toff command has been previously issued.

go
Start protocol. After the protocol finishes processing of the script file will resume after the go command.

end
Terminate script processing. This should be the last command in every script file.

init begin
init end
The `init begin` and `init end` commands can be used to mark a sequence of script commands as being initialization commands. Any commands appearing after a `init begin` and before the next `init end` commands will be treated as initialization commands.

Table 24–1 lists special control sequences and their interpretation which can be used with script files.

Table 24–1 PhoneNet Command Script Control Sequences

Sequence	Interpretation
\r	carriage return (ASCII 13)
\n	newline or line feed (ASCII 10)
\f	form feed (ASCII 12)
\t	tab (ASCII 9)
\ddd	ASCII value ddd (in octal)
\\	backslash character (ASCII 92)
\b	break
\x	delay one second (only for xmit command)

Comment lines are allowed in script files. Comment lines are any lines that have an exclamation point, `!`, in column one.

All characters received and transmitted are logged in the dial protocol transaction log.

A typical script file contains commands to initialize a modem, dial a number, make a connection with a remote system and tell the remote the channel to use. An annotated sample script file is shown in Example 24–4 below.

Example 24–4 Sample PhoneNet Script

```
! p_script.sample - Annotated script to establish dial-out connection
!
! To actually use this file as a dialing script, remove text
! enclosed in parentheses. Comment lines (those beginning with
! "!" can remain or they can be removed if desired.
!
! SECTION 1 - Tell your modem to call a number. Note that this
!             section is modem-dependent. You will need to replace
!             the example dialog of xmit/recv commands with the
!             correct ones for your modem's user interface.
!
init begin              (begin script command which ready the device)
xmit "\r\r"             (send carriage returns - get this modem's attn)
init end                (end script commands which ready the device)
recv "$" 15             (wait 15 sec. for your modem to answer (a $))
xmit "\xK"              (your reply - tell it to dial a number)
recv "NUMBER" 15        (wait 15 sec. for your modem's number prompt)
xmit "\x16175551212\r" (give number - remember 1 and area code)
recv "ON-LINE" 30       (wait 30 sec - for modems that report carrier)
```

Example 24–4 Cont'd on next page

Example 24–4 (Cont.) Sample PhoneNet Script

```
!
! SECTION 2 - Log in to the remote machine (e.g., RELAY.CS.NET).
!             if the remote system is also running PMDF, then the account
!             to log in is the PMDF server account, usually called "PMDF"
!
xmit "\r\x\r"           (send carriage returns - alert remote machine)
recv "ogin:" 30         (wait 30 sec for Login (or login) prompt)
xmit "siteacct\r"       (transmit your account name to remote machine)
recv "assword:" 30      (wait for Password (or password) prompt)
xmit "mypasswd\r"       (transmit password - remember carriage return)
recv "annel:" 120       (wait for Channel (or channel) prompt)
xmit "channelname\r"    (transmit remote's channel name)
go                      (starts the PhoneNet session)
end                     (hangs up and ends the script)
```

Additional example PhoneNet script files can be found in Section 26.4.1.4.

24.6 Backoff retries for undeliverable messages

Note: The following discussion does not apply to channels marked with either the `slave` or `periodic` channel keywords.

As described in Section 1.4, when a message is first enqueued, an immediate message delivery job attempts to deliver the message.[3] Should that delivery attempt fail, then the message will either be returned to the sender or retained for subsequent delivery attempts if, respectively, the failure was permanent or temporary in nature. When the message is retained, periodic delivery jobs will attempt to redeliver any messages which have yet to be delivered.

For some channels, however, it can not be deemed desirable to wait for the next periodic delivery job before another delivery attempt is made. With the BACKOFF channel option and mapping table, it is possible to schedule an immediate message delivery job to attempt a subsequent delivery of a message

Each entry in the BACKOFF mapping table has the format (note the use of the vertical bar character, |)

channel|*n* *time*

where *channel* is the name of the channel, and where *n* is an integer given by the formula

n = (number of delivery attempts so far - 1) / BACKOFF

with BACKOFF the value specified by the BACKOFF option in the channel's option file. *time* is an unsigned integer number of seconds to wait before making another delivery attempt.

[3] Unless, of course, the channel is marked `slave` or `periodic`.

VMS

On OpenVMS, there is an alternate format for specifying the *time* value. If the *time* value is an unsigned integer, it will be interpreted as described above, as the number of seconds to wait before making the next delivery attempt. Alternatively, the value can be specified as an absolute, delta, or combination time specifying when the next delivery attempt should be made.

When the BACKOFF option specifies a value greater than zero, then for each message which cannot be delivered, the BACKOFF mapping table will be consulted. If the table exists and a matching entry is found (*i.e.,* an entry which matches the channel name and *n*), then a delivery job to attempt redelivery for that message will be queued to run at the time specified by *time*. If the table does not exist or no matching entry is found, then no immediate job will be queued and the message will be retried when the next periodic delivery job runs.[4]

For example, suppose that BACKOFF=5 has been specified and that the channel name is dpd_pmdf. Then the following BACKOFF mapping table,

```
BACKOFF

  dpd_pmdf|0       300
  dpd_pmdf|1       600
```

or on OpenVMS the alternate specification

```
BACKOFF

  dpd_pmdf|0       +00:05:00
  dpd_pmdf|1       +00:10:00
```

requests that redelivery attempts should be made every 5 minutes for the first five attempts [*n*=0=(1-1)/5, (2-1)/5, (3-1)/5, (4-1)/5, (5-1)/5], and every 10 minutes for the next five attempts [*n*=1=(6-1)/5, ..., (10-1)/5]. All further attempts will be handled by periodic delivery jobs.

[4] Note that as long as the message is still waiting to be delivered, the periodic delivery jobs will attempt to deliver it irrespective of any pending immediate delivery job.

25 UUCP Channels (OpenVMS and UNIX)

UUCP (UNIX to UNIX Copy Program) is an asynchronous terminal line-based system providing support for file transfer and remote execution between different computer systems. These primitive operations are then used to construct a mail system, which is also, confusingly, known as UUCP. Although UUCP was originally developed for use with the UNIX operating system, versions of UUCP are available for other operating systems, including OpenVMS.

VMS

PMDF for OpenVMS includes support for Encompass UUCP.

PMDF's UUCP support conforms with RFC 976, "UUCP Mail Interchange Format Standard", authored by Mark Horton. A copy of RFC 976 may be found in the RFC subdirectory of the PMDF documentation directory, `PMDF_DOC:[rfc]` on OpenVMS or `/pmdf/doc/rfc` on UNIX.

The following sections describe how to set up PMDF channels for use with UUCP. Familiarity with the basics of UUCP networking is assumed; on OpenVMS systems, familiarity with Encompass UUCP is also assumed.

25.1 Encompass UUCP (VMSNET) Channels for OpenVMS Systems

Encompass UUCP is available from Encompass, the HP User's Group.

25.1.1 Setting Up the Channel

Two or more channels are needed for PMDF for OpenVMS to communicate using Encompass UUCP. A single common channel is used for all incoming messages, no matter from what system they originated. An additional channel is needed for each system connected via UUCP. The incoming message channel is slave-only and should never have any messages queued to it. The outgoing message channels are master-only.

25.1.1.1 Adding the Channel to the Configuration File

The entry for the incoming message channel should look like this (do *not* use a different channel name):

```
vn_gateway uucp slave
vn-gateway
```

The official channel host name (the second line of the channel definition) is arbitrary, but is usually specified as vn-gateway, *e.g.*,

```
vn_gateway uucp slave
vn-gateway
```

Entries for outgoing UUCP message channels will vary depending on the name of the system to which the channel connects. For example, suppose the remote system's official name is YMIR.CLAREMONT.EDU and its UUCP name is simply ymir. A channel definition for this system might be:

```
vn_ymir uucp master
ymir-uucp
ymir.claremont.edu ymir
```

In this case the name of the remote host to which the channel connects is derived from the channel name. When a second channel connecting to the same remote host is needed, it can be defined as follows:

```
vn_second uucp master daemon ymir
ymir-second
ymir.claremont.edu ymir
```

In this case the `daemon` channel keyword has been used to explicitly specify the name of the remote system to which the channel connects.

If the official name and UUCP name are the same, ymir, the entry can be simplified:

```
vn_ymir uucp master
ymir
```

Rewrite rules should be set up to point at the proper outgoing channel using the channel's official host name.

25.1.1.2 Setting Up the Master Program

Once the Encompass UUCP channels have been added to the configuration file, the UUCP master program should be ready to use. No additional log, script or option files are needed; all such information is part of Encompass UUCP.

25.1.1.3 Setting Up the Slave Program

The `uucp_slave` program is used as the `rmail` image. The file that is named `UUCP_BIN:uuxqt_dcl.com` must be edited to add an `rmail` command that invokes `uucp_slave`. As of Encompass UUCP V1.1, the change consisted of removing all the code between the labels `DO_RMAIL:` and `FAILED_RMAIL:` and replacing it with the following:

```
$       on warning then goto failed_rmail
$       pmdf_privilege_save = f$setprv("BYPASS, CMKRNL")
$       rmail = "$PMDF_EXE:uucp_slave.exe"
$       define/user SYS$INPUT 'infile'
$       rmail "''addressee'"
$       delete 'infile';0
$       delete 'xfile'
$       pmdf_privilege_save = f$setprv(pmdf_privilege_save)
$       goto read_loop
```

As of Encompass UUCP V2.0, you would have to change

```
$       rmail = "$PMDF_ROOT:[EXE]uucp_slave.exe"
```

To

```
$       rmail = "$PMDF_EXE:uucp_slave.exe"
```

and also add the following line to your `UUCP_CFG:control.` file.

```
UUCP_UUXQT_DCL_RMAIL_PMDF  "YES"
```

Note: The account under which remote UUCP connections log in to your system must be capable of reading and writing files in the PMDF queue and PMDF log directories, (*i.e.*, in the directories `PMDF_QUEUE:` and `PMDF_LOG:` on OpenVMS) for proper operation. Protections or privileges should be set to allow this access.

25.1.2 Log Files

Various log files are created during the operation of the UUCP channels. All PMDF-specific log files are kept in the PMDF log directory, (*i.e.*, in the directory `PMDF_LOG:` on OpenVMS).

While running, the `uucp_master` program creates a log file, *x*`_master.logfile` where *x* is the channel name. *x*`_master.logfile` logs each message as it is queued to the UUCP system. This file is in the same format as PhoneNet log files. (See Section 24.4.)

Operation of the `uucp_slave` program creates a log file called `rmail.logfile`. This file is also in the same format as PhoneNet log files.

25.1.3 Returning Undelivered Messages

PMDF automatically returns undeliverable messages after a certain amount of time has elapsed. (See Section 1.4.4.) However, Encompass UUCP maintains its own queues for files, so it is possible for messages to get stuck in the UUCP queues where PMDF's regular message return job cannot see them.

Thus, an additional periodic batch job is needed to return undeliverable UUCP messages. This batch job operates in the same way as PMDF's regular message return job except that it scans the UUCP queues and not the PMDF queues. This job is controlled by the command file `PMDF_COM:return_vn.com`.

25.1.4 Starting the Message Return Batch Job

The UUCP message return job must be started manually. Log in to the account under which this job should run (the account must have access to both the PMDF and UUCP queue and log directories) and execute the `PMDF_COM:return_vn.com` command file from the terminal. This action should run the `return_uucp` program (it is a good idea to make sure it is working properly) as well as submitting the periodic batch job.

The `pmdf_submit_jobs.com` procedure, described in the section on post-installation tasks in the OpenVMS edition of the *PMDF Installation Guide*, is capable of checking to make sure the periodic Encompass UUCP message return job is running. To enable this feature, make the following logical name definition:

```
$ DEFINE/SYSTEM PMDF_DO_RETURN_VN 1
```

This logical should be defined before `pmdf_submit_jobs.com` is run during system startup. A convenient way to ensure that this is logical is defined when your system reboots is to put the logical definition in a `PMDF_COM:pmdf_site_startup.com` file you provide; the use of such a file for site specific definitions is adiscussed in the *PMDF Installation Guide, OpenVMS Edition*.

25.1.5 Deinstalling the Encompass UUCP Mailer

Having made all necessary changes, the mailer supplied with Encompass UUCP should be deinstalled; users will be confused as to which to use otherwise. In any case, incoming mail will use PMDF, not the UUCP mailer. To deinstall the UUCP mailer, edit `UUCP_BIN:uucp_systartup.com` to remove all references to MAIL$PROTOCOL_* logicals, remove the line to reinstall `mail.exe` with privileges (this is not needed by the PMDF interface to VMS MAIL) and remove the line that installs the image `UUCP_BIN:uucp_mailshr`.

25.2 UUCP Channels for UNIX Systems

Tru64 UNIX and Solaris support the HoneyDanBer version of UUCP. Refer to the guide *Configuring Your Network Software* to set up UUCP on your system.

25.2.1 Setting Up the Channel

Two or more channels are needed for PMDF on UNIX to communicate using UUCP. A single common channel is used for all incoming messages, no matter from what system they originated. An additional channel is needed for each system connected via UUCP. The incoming message channel is slave-only and should never have any messages queued to it. The outgoing message channels are master-only.

25.2.1.1 Adding the Channel to the Configuration File

The entry for the incoming message channel should look like this (do *not* use a different channel name):

```
uucp_gateway uucp slave
uucp-gateway
```

Entries for outgoing UUCP message channels will vary depending on the name of the system to which the channel connects. For example, suppose the remote system's official name is YMIR.CLAREMONT.EDU and its UUCP name is simply ymir. A channel definition for this system might be:

```
uucp_ymir uucp master
ymir-uucp
ymir.claremont.edu ymir
```

In this case the name of the remote host to which the channel connects is derived from the channel name. When a second channel connecting to the same remote host is needed, it can be defined as follows:

```
uucp_second uucp master daemon ymir
ymir-second
ymir.claremont.edu ymir
```

In this case the `daemon` channel keyword has been used to explicitly specify the name of the remote system to which the channel connects.

If the official name and UUCP name are the same, ymir, the entry can be simplified:

```
uucp_ymir uucp master
ymir
```

Rewrite rules should be set up to point at the proper outgoing channel using the channel's official host name.

25.2.1.2 Setting Up the Master Program

Once the UUCP channels have been added to the configuration file, the UUCP master program should be ready to use. No additional log, script or option files are needed.

25.2.1.3 Setting Up the Slave Program

The PMDF `uucp_slave` program is used to replace the `rmail` program on UNIX. You should rename the original `rmail` program to, *e.g.*, `rmail.org`, and create a symbolic link that links `rmail` to `/pmdf/bin/uucp_slave` as follows:

```
# cd /usr/bin
# mv rmail rmail.org
# ln -s /pmdf/bin/uucp_slave rmail
```

25.2.2 UUCP Channel Option File

An option file may be used to control characteristics of a UUCP channel. Such an option file must be named *x*_option, where *x* is the name of the channel, and stored in the PMDF table directory.

25.2.2.1 Format of the Option File

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

```
option=value
```

value may be either a string or an integer, depending on the option's requirements. If the option accepts an integer value a base may be specified using notation of the form *b*%*v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

25.2.2.2 Available Options

UUCP channels have the following option:

COMMAND_FLAGS (string)
This option may be used to specify command flags to be passed to the uux invocation command, *e.g.*,

```
COMMAND_FLAGS=-gC
```

Multiple flags may be specified, separated with spaces.

25.2.3 Log Files

Various log files are created during the operation of the UUCP channels. All PMDF-specific log files are kept in the PMDF log directory, (*i.e.*, in the directory /pmdf/log on UNIX).

While running, the uucp_master program creates a log file, *x*_master.logfile where *x* is the channel name. *x*_master.logfile logs each message as it is queued to the UUCP system. This file is in the same format as PhoneNet log files. (See Section 24.4.)

Operation of the uucp_slave program creates a log file called rmail.logfile. This file is also in the same format as PhoneNet log files.

25.2.4 Returning Undelivered Messages

PMDF automatically returns undeliverable messages after a certain amount of time has elapsed. (See Section 1.4.4.) However, UUCP maintains its own queues for files, so it is possible for messages to get stuck in the UUCP queues where PMDF's regular message return job cannot see them.

Thus, an additional periodic `cron` job is needed to return undeliverable UUCP messages. This job operates in the same way as PMDF's regular message return job except that it scans the UUCP queues and not the PMDF queues. This job is scheduled by the `cron` daemon.

25.2.5 Starting the Message Return cron Job

The UUCP message return job should be scheduled by `cron`. To submit commands to the `cron` daemon, first become user `pmdf`:

```
# su pmdf
```

To edit the `crontab` entries, issue the command

```
$ crontab -e
```

and use the editor thus invoked to add an entry such as the following:

```
30 1 * * * /pmdf/bin/return_uucp </pmdf/log/return_uucp.log-`/pmdf/bin/unique_id` 2>&1
```

The example entry shown above would be used to run the UUCP return job at 1:30 am and create the log file `/pmdf/log/return_uucp.log-`*uniqueid*, where *uniqueid* will be a unique string disambiguifying the file name, allowing for multiple versions of the file. The first value specifies the minutes after the hour, and the second value specifies the hour — you may wish to specify other values according to the needs of your site. You should use the `return_uucp` shell script as shown above, which itself calls the program `/pmdf/bin/return_uucp`, rather than the UUCP cleanup command, since `return_uucp` will honor the `notices` channel keyword and understand the MIME format of the messages.

26 Other Channels

This chapter contains descriptions of the following channels:

- the addressing channel (see Section 26.1) ,
- the bitbucket channel (see Section 26.2) ,
- the defragmentation channel (see Section 26.3) ,
- the pager channel (see Section 26.4) ,
- the pipe channel (see Section 26.5) ,
- the printer channel (see Section 26.6) ,
- the reprocessing and processing channels (see Section 26.7) ,
- the "generic" SMTP channel (see Section 26.8) , and

Note that some other miscellaneous channels can be found described in other chapters, such as:

- the conversion channel, described in Section 22.1,
- the directory channel, described in Section 3.2,
- the filter_discard channel, described in Section 16.2.5, and
- the mailserv channel, described in Section 4.3.

26.1 Addressing Channels

The addressing channel extracts addressing information from within a message body, constructs a To: address list from the extracted information, and then remails the message, less the addressing information, to the constructed To: address list. Use of such a channel is not the preferred way of handling mail but is often the only way by which some mail systems (*e.g.,* PROFS with its eight-by-eight naming space) can interoperate with other mail systems and networks.

The addressing channel can be used in conjunction with the queue to e-mail symbiont in order to allow applications, such as word processors, to "print" documents directly for transmission as FAXes. This functionality even works with remote printing clients which have access to OpenVMS print queues (*e.g.,* Pathworks, LPD clients, *etc.*).

Note: If you use an option file with the addressing channel, then you must specify in the option file all of the addressing channel commands which you will use.

26.1.1 Channel Operation

When the addressing channel processes a message, it scans, from top to bottom, the body of the message for addressing information. Special commands are used to specify addressing information. The rules used for parsing a message are:

1. The addressing information section begins when the first legal addressing command is encountered; any portion of the message body preceding the addressing information is discarded and not included in the subsequent message.
2. Whitespace between addressing commands is ignored; (except see rule 3, below).
3. At least one blank line must separate a printer address (block) from other addresses.
4. The addressing portion of the message is terminated with a stop or end command or when something other than a legal addressing command is encountered; the portion of the message body following the addressing information is the only part of the message body to be mailed to the To: address list.

The syntax of an addressing command is as follows:

```
:command-name:value
```

where *command-name* is the name of the command and *value* is the addressing information being specified with the command. A delimiter character other than a colon, `:`, can be chosen with the DELIMITER channel option. See Section 26.1.3.2.

Command names are case insensitive and can include punctuation or spaces or both. The default command names are

- A From: address and Subject:, Comments:, and X-PS-Qualifiers: header lines can be specified with the `From`, `Subject`, `Comments`, `PS Qualifiers` commands. These commands are primarily intended for use with files routed to an addressing channel via a queue to e-mail symbiont (*e.g.,* PostScript documents printed from a word processor for FAXing via PMDF-FAX).[1]
- Regular mail addresses are specified using any of the synonymous commands `To`, `Cc`, `Bcc`, `e-mail`, `email`, `PMDF`, or `RFC822`.
- Use the "`delivery receipt`" command to specify an address to which to send delivery receipts. Delivery receipts will be requested of all recipients following this command. Use the command "`no delivery receipt`" to suppress delivery receipts from subsequently specified recipients. Note that `report*` channel keywords can affect the interpretation of such delivery receipt requests.
- Printer addresses are built up using the commands shown in Table 26–1.

Table 26–1 Addressing Channel Printer Commands

Command name	Printer channel AVPL equivalent
`Recipient's name`	AT
`Mail stop`	MS
`Printer domain name`	no equivalent AVPL item

[1] If you want to disable the use of the `From` command, set up an option file which establishes all of the default commands except for the `From` command. See Section 26.1.3.2.

Table 26–1 (Cont.) Addressing Channel Printer Commands

Command name	Printer channel AVPL equivalent
`Recipient's organization`	O
`Recipient's address`	OU
`P1–P8`	P1–P8
`Initialization file`	SETUP
`Recipient's telephone number`	TN
`Username`	USERNAME

In Table 26–1, the command names are listed in the first (left) column and for each command, the portion of a printer channel address which it represents is shown in the second column. The addressing channel will automatically quote items of a printer address which require quoting (*e.g.,* the symbols /, $, and =). See Section 26.6.2 for further information on printer channel addressing conventions.

In building up a printer address, the printer domain name (*e.g.,* printer.example.com) must be specified. If the DEFAULT_DOMAIN channel option has been used to establish a default domain for use with FAX or printer addresses, then the printer domain name can be omitted, in which case the default domain name will instead be used.

- The addressing information can be terminated with a `stop` or `end` command, *e.g.*:

```
:stop:
```

The use of either of these two commands is optional in most cases. Only when PostScript files are being routed to an addressing channel via a queue to e-mail symbiont is their use mandatory.

26.1.2 Examples

Figure 26–1 illustrates a message which is to be addressed to

```
smith@vax.example.com
"Bob Jones" <jones@stateu.edu>
"/fn=621-5319/at=Mrocheck/o=Example, Inc./ou=250 South St./"@text-fax.example.com
"/tn=Beckett/o=Harvey Mudd College/ou=Academic Computing/"@printer.example.com
```

Figure 26–1 Addressing Channel Sample

```
This initial text appearing before the addressing commands
will be discarded.
❶

:e-mail: smith@vax.example.com ❷
:e-mail: "Bob Jones" <jones@stateu.edu> ❸
```

Figure 26–1 Cont'd on next page

Figure 26–1 (Cont.) Addressing Channel Sample

```
:FAX telephone number:     621-5319 ❹
:Recipient's name:         Mrochek
:Recipient's organization: Example, Inc.
:Recipient's address:      250 South St.
:FAX gateway address:      text-fax.example.com

:Recipient's name:         Beckett ❺
:Recipient's organization: Harvey Mudd College
:Recipient's address:      Academic Computing
:Printer domain name:      printer.example.com
:stop:

Only this text and any subsequent text will be mailed to the addressees
specified above. ❻
```

❶ Any portion of the message which appears before the addressing information is discarded (*i.e.,* will not be forwarded on with the remainer of the message).

❷ The first To: address, smith@vax.example.com.

❸ The second To: address, jones@stateu.edu.

❹ The third To: address which is a PMDF-FAX address. Note the required blank line separating the FAX address block from the e-mail addresses.

❺ The fourth and final To: address which is a printer channel address. Note that the preceding blank line is required.

❻ The portion of the message body which will be sent to the four To: addressees.

26.1.3 Setting Up the Channel

The following two sections document how to add an addressing channel to your configuration and, if desired, how to customize the command names recognized by the channel.

Note: More recent versions of the PMDF-MTA configuration utility automatically generate an addressing channel in the PMDF configuration. If you have configured PMDF using a recent version of the configuration utility and hence already have an addressing channel in your configuration, then the steps in Section 26.1.3.1 have already been performed for you.

26.1.3.1 Adding the Channel to the Configuration File

To set up an addressing channel, you must add rewrite rules and a channel block definition to the PMDF configuration file.

The first step is to pick a domain name for the addressing channel; for example, addressing.example.com. Mail to be processed by the channel should then be addressed to *x*@addressing.example.com. The value of *x* is irrelevant — it is never used by the channel and, at present, simply discarded. After selecting a domain name, add rewrite rules for it to the top of the configuration file

```
addressing                      $U%domain-name@ADDRESSING-DAEMON
domain-name                     $U%domain-name@ADDRESSING-DAEMON
```

where *domain-name* is the domain name selected for the addressing channel. For example, if the domain name selected is addressing.example.com, then the rewrite rule would appear as

```
addressing.example.com          $U%addressing.example.com@ADDRESSING-DAEMON
```

Next, add the channel block

```
address
ADDRESSING-DAEMON
```

to the end of the configuration file. Be sure to include a blank line before the first line of the channel block.

If desired, additional addressing channels can be added. They should have names beginning with "address_" and not exceed a length of 32 characters.

Note: If you are part of a TCP/IP network, then you can want to add the addressing channel's domain name (or names) you selected to your DNS using MX records. This will allow other machines to route mail to the addressing channel.

26.1.3.2 Option Files

An option file can be used to alter the addressing commands recognized by the addressing channel. Option files are stored in the PMDF table directory and have names of the form `x_option`, where *x* is the name of the channel to which the option file applies. (In most instances, the file will be `PMDF_TABLE:address_option.` on OpenVMS, or `/pmdf/table/address_option` on UNIX, or typically `C:\pmdf\table\address_option` on NT.)

If you supply an option file, then you must define all commands which are to be accepted. This is even the case if you are just supplying an option file to set some non-command related parameter. This behavior is intentional; it allows commands to be disabled.

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

```
option=command-name
```

where the valid values for *option* are given in Table 26–2 and *command-name* is a string specifying the name of a command to establish. Synonymous commands can be established by using the same option in the options file; *e.g.,* EMAIL in Example 26–1.

Table 26–2 Addressing Channel Options

Option	Usage
AFTER	FAX address AFTER attribute
AT	FAX & printer address AT attribute
AUTH	FAX address AUTH attribute
COVER	FAX address COVER attribute
COMMENTS	RFC 822 Comments: header line
DEFAULT_DOMAIN	Domain name to use when none specified for a FAX or printer address; by default no default domain name is established
DELIMITER	Single character used to delimit command names; *e.g.,* the colon, :, in ":stop:"
DELRCPT	Address to which to send delivery receipts
DMN	FAX & printer domain name
EMAIL	Regular RFC 822 To: address (synonym for EMAIL_TO)
EMAIL_BCC	Regular RFC 822 Bcc: address
EMAIL_CC	Regular RFC 822 Cc: address
EMAIL_TO	Regular RFC 822 To: address
FN	FAX address FN attribute
FROM	RFC 822 From: address
FSI	FAX address FSI attribute
MIME_MODE	Control whether the addressing channel will process commands from MIME encoded message parts
MS	Printer address MS attribute
NDELRCPT	Suppress delivery receipt requests
O	FAX & printer address O attribute
OU	FAX & printer address OU attribute
P1–P8	Printer address P1–P8 attributes
PS_QUALIFIERS	RFC 822 X-PS-Qualifiers: header line
RP_MODE	Control whether the addressing channel operates in a mode compatible with parsing MIME application/remote-printing message parts
SETUP	FAX address SETUP attribute
SFN	FAX address SFN attribute
STN	FAX address STN attribute
STOP	String to recognize as a stop or end command
SUBJECT	RFC 822 Subject: header line
TN	FAX & printer address TN attribute

Table 26–2 (Cont.) Addressing Channel Options

Option	Usage
TTI	FAX address TTI attribute
USERNAME	Username to associate with printer channel print jobs

The example option file given in Example 26–1 establishes the default command set used by the addressing channel. A copy of this file, `address_option.sample`, can be found in the PMDF table directory, *i.e.*, `PMDF_TABLE:address_option.sample` on OpenVMS or `/pmdf/table/address_option.sample` on UNIX.

One of the most useful options is the DEFAULT_DOMAIN option. To prevent, for instance, users from having to specify

```
:FAX gateway address: ps-fax
```

in messages, you can specify

```
DEFAULT_DOMAIN=ps-fax
```

in an option file. Remember that the existence of the option file clears all commands and you must explicitly define each command in the option file. If you want to use the default commands but specify a default domain, then simply copy the sample file, `address_option.sample`, to an option file with the correct name (*e.g.,* `address_option`), and then add to that file the DEFAULT_DOMAIN setting.

26.2 Bitbucket Channel

As the name itself suggests, the bitbucket channel simply deletes any message enqueued to it. Indeed, messages that match the bitbucket channel are instantly deleted, without even being written to a bitbucket disk area. (In particular, note that no `PMDF_QUEUE:[bitbucket]*` (OpenVMS) or `/pmdf/queue/bitbucket/*` (UNIX) or `C:\pmdf\queue\bitbucket\*` (NT) message files are created—a message is simply discarded immediately once PMDF sees that it matches the bitbucket channel.)

26.2.1 Configuration

A bitbucket channel must be added to PMDF's configuration file in order to be used. The entry should have the form:

```
bitbucket
BITBUCKET-DAEMON
```

Example 26–1 Example Addressing Channel Option File

```
! *** If you make your own option file, then you must specify _ALL_ the
! *** addressing commands which you use!
!
DELIMITER=:
MIME_MODE=0
RP_MODE=0
EMAIL_TO=To
EMAIL_CC=Cc
EMAIL_BCC=Bcc
EMAIL_TO=e-mail
EMAIL_TO=email
EMAIL_TO=PMDF
EMAIL_TO=RFC822
COMMENTS=Comments
PS_QUALIFIERS=PS Qualifiers
AFTER=Transmit after
AT=Recipient's name
AUTH=Authorization code
COVER=Cover page
DELRCPT=Delivery receipt
DMN=FAX gateway address
DMN=Printer domain name
FN=FAX telephone number
FROM=From
FSI=FAX modem's telephone number
MS=Mail stop
NDELRCPT=No delivery receipt
O=Recipient's organization
OU=Recipient's address
P1=P1
P2=P2
P3=P3
P4=P4
P5=P5
P6=P6
P7=P7
P8=P8
SETUP=Initialization file
SFN=My FAX telephone number
STN=My telephone number
STOP=end
STOP=start
STOP=stop
SUBJECT=Subject
TN=Recipient's telephone number
TTI=My organization
USERNAME=Username
USERNAME=User
```

And rewrite rules should be added for some pseudodomain name(s) which will be used when directing messages to the bitbucket channel; for instance,

```
bitbucket                   $U%bitbucket.localhostname@BITBUCKET-DAEMON
bitbucket.localhostname     $U%bitbucket.localhostname@BITBUCKET-DAEMON
```

where *localhostname* is the name of the local PMDF system.

26.3 Defragmentation Channel

The MIME standard (RFC 2046) provides the message/partial content type for breaking up messages into smaller parts. This is useful when messages have to traverse networks with size limits. Information is included in each part so that the message can be automatically reassembled once it arrives at its destination.

The `defragment` channel keyword and the defragmentation channel provide the means to reassemble messages in PMDF. When a channel is marked `defragment` any message/partial messages queued to the channel will be placed in the defragmentation channel queue instead. The defragmentation channel maintains a database which is used to match the parts of each message up with each other. Once all the parts have arrived the message is rebuilt and sent on its way.

All channels that perform local delivery or send messages on to networks that cannot deal with fragmented messages should be marked with the `defragment` channel keyword. The `defragment` channel keyword will have no effect unless a defragmentation channel is also defined.

A defragmentation channel is produced automatically by the PMDF configuration generator.

26.3.1 Defragmentation Channel Definition and Rewrite Rules

If your configuration was generated by the PMDF configuration utility[2] (PMDF V4.1 or later), then you do not need to add a defragmentation channel to your configuration: this was done automatically for you by the configuration utility. See the appropriate edition of the *PMDF Installation Guide* for instructions on using the configuration utility.

The first step in installing a defragmentation channel is to insert the channel entry in PMDF's configuration file. The entry should have the form:

```
defragment
DEFRAGMENT-DAEMON
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the defragmentation channel. To do this, add the rewrite rules

[2] The web based configuration utility, or the command line utility `pmdf configure` (UNIX) or PMDF CONFIGURE (OpenVMS) or in older versions of PMDF for OpenVMS, one of the command procedures `configure.com` or `access_configure.com`

```
defragment                  $U@defragment.localhostname@DEFRAGMENT-DAEMON
defragment.localhostname    $U@defragment.localhostname@DEFRAGMENT-DAEMON
```

where *localhostname* should be replaced by the name of the local host. Once this is done, an address of the form

user%host@defragment.*localhostname*

will be routed through the defragmentation channel. (Sending anything other than a message/partial message to the defragmentation channel causes the channel to simply requeue the message for normal delivery.)

26.3.2 Defragmentation Channel Retention Time

Messages are retained in the defragment channel queue only for a limited time. When one half of the time before the first nondelivery notice is sent has elapsed, the various parts of a message will be sent on without being reassembled. This choice of time value eliminates the possibility of a nondelivery notification being sent about a message in the defragment channel queue.

The `notices` channel keyword controls the amount of time that can elapse before nondelivery notifications are sent, and hence also controls the amount of time messages are retained before being sent on in pieces. Set the `notices` keyword value to twice the amount of time you want to retain messages for possible defragmentation. For example, a `notices` value of 4 would cause retention of message fragments for two days:

```
defragment notices 4
DEFRAGMENT-DAEMON
```

26.4 Pager Channels

Pager channels can be used to route e-mail messages to remote paging switches for transmission to alpha-numeric page receivers. The pager channel will operate with paging switches which use the TAP dialup protocol (Telelocator Alphanumeric input Protocol; referred to as IXO). A dialup modem is used to communicate with remote paging switches.

As necessary, the pager channel will automatically fragment large messages into multiple pages so that no one page will exceed paging switch and page receiver size limits.[3] The pager channel can also generate status reports and delivery acknowledgements.

TAP is primarily supported in North America. It is, however, finding a foot hold in other countries. For instance, the PMDF pager channel is reported to work in Ireland where TAP is also used. A subset of TAP, called PET (Pager Entry Terminal), is supported in Australia and New Zealand. In Australia, The University of Melbourne has obtained certification from Telecom Australia for the use of PMDF's pager channels with Telecom

[3] This fragmentation mechanism is, by necessity, separate from PMDF's general message fragmentation facilities.

Australia PET switches. See Section 26.4.1.6 for important directions on configuring pager channels for use with PET switches.

26.4.1 Setting Up the Channel

There are four steps in setting up a pager channel: (1) adding the necessary channel blocks and rewrite rules to the PMDF configuration file, (2) setting up a PAGER mapping table, (3) setting up a modem script, and (4) setting up a channel option file, if necessary. A fifth and optional step is to set up a directory channel. With a directory channel, pager addressing can be simplified. For instance, rather then sending mail to /id=1234/msglen=200/@mci.pager.example.com, addresses like andy@pager.example.com can be used. This obviates, from the user's perspective, the need to remember pager numbers everytime a page is to be sent.

26.4.1.1 Before You Start

Before you begin setting up a pager channel, you should try to obtain the following information from your paging service provider:

1. The phone numbers, preferably local, for each dial-up paging switch you plan to use. Attempting to obtain these numbers can be quite frustrating: if you do not use the proper terminology, the customer service representatives will not know what you are talking about. Unfortunately, every service provider seems to have their own, unique terminology.

   You are asking for the phone number of an auto-answer, asynchronous modem. When you dial it, you should hear a modem tone. Some possible ways of inquiring after this number include:

   a. Ask for the telephone number for "alpha-numeric paging".

   b. Ask, "If I had a Motorola AlphaMate, what telephone number would I use?" An AlphaMate is an intelligent terminal with an internal modem that some paging companies will provide for you to send alpha-numeric pages.

   c. Ask for the "paging modem telephone number".

   If none of this works, ask to talk to a technician and repeat the questions.

2. The dialup modem characteristics (baud rate, *etc.*). It is very unlikely that your paging provider will be able to tell you this information, but if you like hitting your head up against a brick wall you can try asking them anyhow. We have not found any paging switches that will not accept 300 baud, even parity, with 7 data bits. We have seen paging switches which will handle 1200 and 2400 baud. We have also seen paging switches which will accept "no parity" even though they send even parity. We have also heard of paging switches which will claim to connect at, say 1200 or 2400 baud, but will nonetheless transmit at 300 baud.

3. The maximum number of bytes per page accepted by the switch. Do not confuse this limit with that imposed by a page receiver. The limit imposed by the switch is handled

through the MAX_PAGE_SIZE channel option; the limit imposed by individual pager units is controlled on a per address basis with the PAGLEN address attribute.

If the paging provider seems unsure of this, or if your longer pages don't work, you will have to determine this value experimentally. You can find that you have to limit your pages to a few bytes less than what the provider thinks the maximum is.

Note that this information can generally be determined by trial-and-error: enable `master_debug` for the pager channel and send some messages of varying length. Then look in the file `pager_`*`name`*`_master.log` (where *`name`* is the specific part of the name of a pager channel) in the PMDF log file directory and see if the switch rejects the page.[4] If it does reject pages, then the values set with the MAX_PAGE_SIZE or MAX_BLOCKS_PER_PAGE channel options can need to be reduced. Many switches only accept single pages of lengths less than 200±100 bytes. The pager channel will automatically break messages which exceed MAX_PAGE_SIZE into multiple pages, each page requiring no more than MAX_BLOCKS_PER_PAGE data blocks.

26.4.1.2 Adding the Channel to the Configuration File

First choose domain names (*i.e.,* host names) to associate with each paging switch you intend to use. These domain names will be used when addressing a message to a particular paging switch. For instance, suppose the local domain is EXAMPLE.COM and mail is to be routed to one of two paging switches operated by Pacific Telephone and MCI. Then appropriate domain names might be pactel.pager.example.com and mci.pager.example.com.

After choosing domain names, *`domain-name-1`*, *`domain-name-2`*, ..., add to the PMDF configuration file channel block entries of the form

```
pager_x1
domain-name-1

pager_x2
domain-name-2

.
.
.
```

Here *`x1`*, *`x2`*, ... are unique strings, each less than 26 characters in length, which serve to uniquely identify each instance of a pager channel. Be sure to leave a blank line before and after each channel block you add to your configuration file.

Continuing with our example from above, the channel blocks might appear as

```
pager_pactel
pactel.pager.example.com

pager_mci
mci.pager.example.com
```

[4] Read Section 26.4.1.7 before beginning any trials. Note especially that some switches merely time out when presented with too long of a page!

After adding the channel block, go to the top of the configuration file and add rewrite rules of the form:

```
domain-name-1          $U@domain-name-1
domain-name-2          $U@domain-name-2
.
.
.
```

For instance, in the context of our example, the following rewrite rules should be added

```
pactel.pager.example.com          $U@pactel.pager.example.com
mci.pager.example.com             $U@mci.pager.example.com
```

Note: If you are part of a TCP/IP network, then you can want to add the pager domain names you selected to your DNS using MX records. This will allow other machines to route mail to your pager channels.

26.4.1.3 PAGER Mapping

The next step in setting up one or more pager channels is to add a mapping table named PAGER to the mapping file. This table serves three purposes: it specifies which message header lines to include in a page, how to abbreviate those header lines, and how to abbreviate the body of a message (*i.e.,* remove superfluous spaces, abbreviate words, *etc.*). Some familiarity with the use of the mapping file is helpful at this point; see Chapter 5.

The name of the mapping table should either be PAGER or PAGER_*x* where *x* is the name of the pager channel (*e.g.,* PAGER_pager_pactel or PAGER_pager_mci). Each time a pager channel runs, it will first attempt to load the mapping table PAGER_*x* or, if that fails, it will then attempt to load the table PAGER. If neither table can be loaded, then the entire message will be sent in its entirety.[5]

Two types of entries can be made in the mapping table. However, before explaining the format of those entries, let it be made clear that an understanding of how to use the mapping file is essential in order to understand how to construct and use these entries. A sample mapping table is given after the description of these two types of entries. This sample table, which exists as the file `pager_table.sample` in the PMDF table directory, will probably meet most site's initial needs and can simply be included into the mapping file as follows: (a) copy the file to `pager_table.txt`, (b) set this new file to be world readable, and (c) include a file reference to it in the PMDF mapping file.[6] That is, to the PMDF mapping file add an entry such as on OpenVMS

```
<PMDF_TABLE:pager_table.txt
```

or on UNIX

[5] It will, however, be truncated if its length exceeds the MAX_MESSAGE_SIZE or MAX_PAGES_PER_MESSAGE parameters or any sizes specified with either the MSGLEN or MAXPAG address attributes.

[6] The mapping file is the file pointed at by the PMDF_MAPPING_FILE logical on OpenVMS, or PMDF tailor file option on UNIX, or Registry entry on NT, so usually `PMDF_TABLE:mappings.` or `/pmdf/table/mappings` or `C:\pmdf\table\mappings`, respectively. If this file does not already exist, then create it now. This file must be world readable. If you have a compiled configuration, then you must recompile and re-install your configuration so that this file, and changes to it, will be seen and used by PMDF. See Chapter 5 for complete details on the mapping file.

```
</pmdf/table/pager_table.txt
```

or on NT

```
<C:\pmdf\table\pager_table.txt
```

Now, the two types of entries are as follows:

1. *Message header entries.* These entries specify which message header lines should be included in a page and how they should be abbreviated or otherwise mapped. Only if a header line is successfully mapped to a string of non-zero length by one of these entries will it be included in the page being generated. Each entry has the format

   ```
   H|pattern       replacement-text
   ```

   If a message header line matches the pattern *pattern* then it will be replaced with the replacement text *replacement-text* using the mapping file's pattern matching and string substitution facilities. The final result of mapping the header line will then be included in the page provided that the metacharacter $Y was specified in the replacement text. If a header line does not match any pattern string, if it maps to a string of length zero, or if the $Y metacharacter is not specified in the replacement text, then the header line will be omitted from the page.

   The two entries

   ```
   H|From:*        F:$0$Y
   H|Subject:*     S:$0$Y
   ```

   cause the From: and Subject: header lines to be included in pages with "From:" and "Subject:" abbreviated as "F:" and "S:". The entries:

   ```
   H|Date:*                        H|D:$0$R$Y
   H|D:*,*%19%%*:*:*               H|D:$0$ $5:$6$R$Y
   ```

   cause the Date: header line to be accepted and mapped such that, for instance, the header line "Date: Wed, 15 Dec 2012 16:13:27 -0700 (PDT)" will be converted to "D: Wed 16:13".

   Very complicated, iterative mappings can be built. Managers who want to set up custom filters will first need to understand how the mapping file works. The "H | " in the right-hand-side of the entry can be omitted, if desired. The "H | " is allowed in that side so as to cut down on the number of table entries required by sets of iterative mappings.

2. *Message body mappings.* These entries establish mappings to be applied to each line of the message body. Each line of the message body will be passed through these mappings before being incorporated into the page being built. These entries take the form:

   ```
   B|pattern       B|replacement-text
   ```

   If a line of the message body matches a pattern *pattern* then it will be replaced with the replacement text *replacement-text*.

   Again, very complicated, iterative mappings can be constructed using this facility. The "B | " in the right-hand-side of the entry can be omitted, if desired.

The file pager_table.sample from the PMDF table directory is shown in Example 26–2. The entries in this example are explained below.

❶ These two entries cause From: and Subject: header lines to be included in a page. From: and "Subject:" are abbreviated as, respectively, F: and S:. Some of the other entries can have further effects on From: and Subject: header lines.

❷ This entry will reduce a From: header line containing a <...> pattern to only the text within the angle brackets. *E.g.,* "F: "John C. Doe" <jdoe@example.com> (Johnny)" will be replaced with "F: jdoe@example.com".

❸ This entry will remove, inclusively, everything inside of a (...) pattern in a From: header line. *E.g.,* "F: "John C. Doe" <jdoe@example.com> (Johnny)" will be replaced with "F: "John C. Doe" <jdoe@example.com>".

❹ This entry will remove, inclusively, everything inside of a "..." pattern in a From: header line. *E.g.,* "F: "John C. Doe" <jdoe@example.com> (Johnny)" will be replaced with "F: <jdoe@example.com> (Johnny)".

❺ This entry will remove, inclusively, everything to the right of an at-sign, @, in a From: header line. *E.g.,* "F: "John C. Doe" <jdoe@example.com> (Johnny)" will be replaced with "F: "John C. Doe" <jdoe".

❻ These four entries remove leading and trailing spaces from lines in the message header and body.

❼ These two entries reduce two spaces to a single space in lines of the message header and body.

❽ These four entries reduce double dashes, periods, exclamation points, and question marks to single occurrences of the matching character. Again, this helps save bytes in a page.

Example 26–2 Sample PAGER Mapping Table

```
PAGER

  H|From:*                          H|F:$0$R$Y  ❶
  H|Subject:*                       H|S:$0$R$Y  ❶
  H|F:*<*>*                         H|F:$1$R$Y  ❷
  H|F:*(*)*                         H|F:$0$2$R$Y  ❸
  H|F:*"*"*                         H|F:$0$2$R$Y  ❹
  H|F:*@*                           H|F:$0$R$Y  ❺
  H|%:$ *                           H|$0:$1$R$Y  ❻
  H|%:*$                            H|$0:$1$R$Y  ❻
  H|%:*$ $ *                        H|$0:$1$ $2$R$Y  ❼
```

```
B|*--*                          B|$0-$1$R  ❽
B|*..*                          B|$0.$1$R  ❽
B|*!!*                          B|$0!$1$R  ❽
B|*??*                          B|$0?$1$R  ❽
B|*$ $ *                        B|$0$ $1$R  ❼
B|$ *                           B|$0$R  ❻
B|*$                            B|$0$R  ❻
```

In Example 26–2, the metacharacter $R is used to implement and control iterative application of the mappings. By iterating on these mappings, powerful filtering is achieved. For instance, the simple mappings to remove a single leading or trailing space (❻) or reduce two spaces to a single space (❼) become, when taken as a whole, a filter which strips *all* leading and trailing spaces and reduces all consecutive multiple spaces to a single space. Such filtering helps reduce the size of each page.

The order of the entries is very important. For instance, with the given ordering, the body of the message From: header line

From: "John C. Doe" <jdoe@example.com> (Naples)

will be reduced to

jdoe

The steps taken to arrive at this are as follows:

1. We begin with the From: header line

   From: "John C. Doe" <jdoe@example.com> (Naples)

   The pattern in the first mapping entry matches this and produces the result

   F: "John C. Doe" <jdoe@example.com> (Naples)

   The $R metacharacter in the result string causes the result string to be remapped.

2. The mapping ❷ is applied to the result string of the last step. This produces:

   F: jdoe@example.com

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

3. Next, the mapping ❺ is applied producing

   F: jdoe

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

4. Next, the mapping ❻ is applied producing

   F:jdoe

   The $R in the mapping causes the entire set of mappings to be re-applied to the result of this step.

5. Since no other entries match, the final result string

   F:jdoe

   is incorporated into the page.

Note that the PMDF TEST/MAPPING utility (OpenVMS) or `pmdf test -mapping` utility (UNIX and NT) can be used to test the PAGER mapping table. For instance, on OpenVMS:

```
$ PMDF TEST/MAPPING/NOIMAGE_FILE/MAPPING_FILE=PMDF_TABLE:pager_table.sample
Enter table name: PAGER
Input string: H|From: "John C. Doe" <jdoe@example.com> (Naples)
Output string: H|F:jdoe
Output flags: [0,1,2,89]
Input string: ^Z
$
```

Or on UNIX:

```
# pmdf test -mapping -noimage_file -mapping_file=/pmdf/table/pager_table.sample
Enter table name: PAGER
Input string: H|From: "John C. Doe" <jdoe@example.com> (Naples)
Output string: H|F:jdoe
Output flags: [0,1,2,89]
Input string: ^D
#
```

Or on NT:

```
C:\> pmdf test -mapping -noimage_file
-mapping_file=C:\pmdf\table\pager_table.sample
Enter table name: PAGER
Input string: H|From: "John C. Doe" <jdoe@example.com> (Naples)
Output string: H|F:jdoe
Output flags: [0,1,2,89]
Input string: ^C
C:\>
```

See Chapter 29 and Chapter 30 for information on using these utilities.

26.4.1.4 Modem Script

To handle the dialout modems used to call paging switches, the pager channel uses PhoneNet's modem handling facilities. Specifically, the pager channel uses the `phone_list.dat` file, described in Section 24.3, and modem script files, described in Section 24.5.

Before writing a modem script, you should manually dial the paging switch with the modem you intend to use. Record the modem commands required to dial the switch and the modem responses generated in the process. These commands and responses will then form the basis for your modem script.

On all platforms, the modem and serial line must be 8 bits, no parity.

For each pager channel, a separate script file is required, stored in a required subdirectory of the PMDF table directory. Each such script file must have a name of the form

`pmdf_root:[table.`*`channel`*`]`*`script`*`_script.` (OpenVMS) or
`/pmdf/table/`*`channel`*`/`*`script`*`_script` (UNIX) or
`C:\pmdf\table\`*`channel`*`\`*`script`*`_script` on NT

where *`channel`* is the name of the pager channel and *`script`* is the script option specified in the `phone_list.dat` file.

Suppose that two pager channels, pager_pactel and pager_mci, are to be set up. Then a phone_list.dat file in the PMDF table directory and two script files need to be set up. A sample OpenVMS phone_list.dat file is shown in Example 26–3, and an analogous sample UNIX phone_list.dat file is shown in Example 26–4.

Example 26–3 Example OpenVMS phone_list.dat File for Two Pager Channels

```
pager_pactel ❶   TTA3: ❷  hayes ❸
pager_pactel ❶   TTA3:     hayes
pager_mci ❹      TTA3:     hayes
pager_mci ❹      TTA3:     hayes
```

Example 26–4 Example UNIX phone_list.dat File for Two Pager Channels

```
pager_pactel ❶   /dev/ttyd0 ❷  hayes ❸
pager_pactel ❶   /dev/ttyd0     hayes
pager_mci ❹      /dev/ttyd0     hayes
pager_mci ❹      /dev/ttyd0     hayes
```

In these examples, the following items should be noted:

❶ Two identical entries (methods) are provided for the pager_pactel channel. If the first method fails, then the second method will be attempted. By providing two identical methods, we ensure that if, for some reason, the first call fails (busy signal, no answer) a second attempt will immediately be made.

❷ The same terminal line, (TTA3: on OpenVMS or /dev/ttyd0 on UNIX), is used by all methods listed. The dialup modem is connected to this terminal line.

❸ The same script prefix name is used by each method. However, two script files, are required. Both are named hayes_script, but one is stored in the pager_pactel subdirectory and the other in the pager_mci subdirectory of the PMDF table directory.

❹ Two identical entries (methods) are provided for the pager_mci channel.

The sample OpenVMS phone_list.dat file shown in Example 26–3 calls for two script files,

pmdf_root:[table.pager_pactel]hayes_script., and
pmdf_root:[table.pager_mci]hayes_script.

The sample UNIX phone_list.dat file shown in Example 26–4 calls for two script files,

/pmdf/table/pager_pactel/hayes_script, and
/pmdf/table/pager_mci/hayes_script.

When the same modem is used for each pager channel, as in Example 26–3 and Example 26–4, the only difference between the individual script files is usually just the phone number to be dialed for each paging switch. A sample script file intended for a Hayes compatible modems is shown in Example 26–5. (A sample script for Racal-Vadic 212a modems is given in Example 26–6.) Nearly all of the script commands shown are modem specific. In writing your own script file, a knowledge of your modem's command language as well as the responses to expect from the modem is essential.

Example 26–5 Example Hayes-compatible Modem Script

```
init begin ❶
xmit "ATV1&M0&N1\r"  ❷
recv "OK" 10  ❸
init end ❹
xmit "\xATDT99500572\r"  ❺
recv "CONNECT" 30  ❻
xmit "\x\r\x"  ❼
go  ❽
xmit "\xATHZ\r"  ❾
end
```

The following notes apply to Example 26–5.

❶ Begin modem initialization.

❷ Issue "verbal" response codes rather than numeric response codes (V1); turn off MNP negotiations (&M0); and tell the modem to negotiate a 300 baud connection (&N1). Your modem might not understand MNP and thus lack the &M0 command. Or, it might use some command other than &M0 to disable MNP negotiations.

❸ Wait up to 10 seconds for the response "OK".

❹ End modem initialization.

❺ Pause one second and then dial the paging switch.

❻ Wait up to 30 seconds for the response "CONNECT".

❼ Pause one second and then send a carriage return. With some paging switches you can need to omit this xmit command.

❽ Connection has been made; allow the pager channel to take over.

❾ Pager channel is done; force the modem to hangup (H) and then reset the modem (Z).

The following notes apply to Example 26–6.

❶ Start of the modem initialization section of the script.

❷ Transmit a control-E followed by a carriage return.

❸ Wait up to 10 seconds for the response "*".

❹ End of the modem initialization section of the script.

❺ Pause one second and then issue the modem's dial command.

❻ Wait up to 10 seconds for the response “UMBER?”.

❼ Pause one second, specify that the modem should establish a 300 baud connection, and then specify the number to be dialed (625-6149).

❽ Wait up to 10 seconds for the modem to echo the phone number to be dialed.

❾ Pause one second and then transmit a carriage return. After the return is received by the modem, the modem will dial the number.

❿ Wait up to 30 seconds for the response “N LINE”.

⓫ Connection has been made; allow the pager channel to take over.

Example 26–6 Example Racal-Vadic 212a Modem Script

```
init begin ❶
xmit "\005\r" ❷
recv "*" 10 ❸
init end ❹
xmit "\xD\r" ❺
recv "UMBER?" 10 ❻
xmit "\x9k6256149\r" ❼
recv "6149" 10 ❽
xmit "\x\r"  ❾
recv "N LINE" 30 ❿
go ⓫
end
```

The following is a fine technical point — don’t worry if you don’t understand this the first time.

VMS

Normally, when the script processing fails, (*e.g.,* a `recv` command times out), the next method, if any, in the `phone_list.dat` file is attempted. However, this defeats the use of modem pools on OpenVMS:[7] for instance, if the initialization of the first device in the pool fails then rather than try any other devices in the pool, the next method will be tried. However, should the script fail because of a post initialization problem (*e.g.,* the dialed number was busy), then attempting the next method rather than the next device in the modem pool is the proper behavior. The `init begin` and `init end` script commands should be used in conjunction with modem pools: they mark one or more sections of the script file as being modem initialization steps. Should the script fail while processing a command in a script section delimited by `init begin` and `init end` , then the pager channel will first try any other devices in the search list. Only after that list of devices has been exhausted, will the next method from the `phone_list.dat` file be tried. Failures which occur in commands outside of an initialization section will result in the next method being tried.

[7] When a modem pool is used on OpenVMS, the device name given by a method in the `phone_list.dat` file is a logical which translates to a list of device names; *e.g.,*

```
$ DEFINE/SYSTEM MODEMS LTA1:,LTA2:,TXA6:,LTA133:
```

26.4.1.5 Option Files

A great number of options can be set with an option file. Option files are used to set run-time options on a per channel basis. Option files are stored in the PMDF table directory and have names of the form *x*_option, where *x* is the name of the pager channel to which the option file applies, *i.e.*, PMDF_TABLE:*x*_option. on OpenVMS, or /pmdf/table/*x*_option on UNIX, or C:\pmdf\table*x*_option on NT.

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

option=value

value can be either a string or an integer, depending on the option's requirements. If the option accepts an integer value, *value*, a base can be specified using notation of the form *b%v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

Comments are allowed. Any line that begins with an exclamation point, !, is considered to be a comment and is ignored. Blank lines are also ignored in any option file.

The available options are:

ACK_SUCCESS (0 or 1)
When the ACKNOWLEDGEMENT option specifies a value greater than zero, both status reports and a final delivery report (success or failure) will be generated. By specifying ACK_SUCCESS=0, any final success delivery report will be suppressed (not sent); the originator of the page will then only receive status reports and any final error report. This option is ignored when ACKNOWLEDGEMENT specifies a value less than unity. To suppress all status reports and success delivery reports (*i.e.,* only send a message when an error occurs), simply specify ACKNOWLEDGEMENT=-1.

ACKNOWLEDGEMENT (integer)
By default, delivery acknowledgements are not generated (ACKNOWLEDGEMENT=-1). However, when this option specifies an integer value greater than or equal to zero, then a delivery acknowledgement will be sent to the message originator after a page has been successfully transmitted to the paging switch *provided that either NOTARY was not used to submit or relay the message to PMDF, or, if NOTARY was used, that delivery reports (DSNs) were requested*.

Acknowledgements are preferentially sent to the delivery receipt address specified by a Delivery-receipt-to: header line. If no such header line is specified, then the address specified by a Reply-to: header line is used. Finally, if no Reply-to: header line is present, then the envelope From: address is used. If the originating message specifies a delivery receipt address of <>, then no acknowledgement will be generated. If USE_REPLY_TO=0 has been specified in the option file, then the Reply-to address will never be used.

If set to a positive integer $n >= 1$ then status reports are sent every nth unsuccessful transmission attempt starting with the first attempt; *i.e.,* on attempts 1, 1+n, 1+2n, 1+3n, ... For instance, if ACKNOWLEDGEMENT=1 then a status report will be sent every attempt until the message is successfully sent, and then, when the message is successfully sent, an acknowledgement will be sent. If, however, ACKNOWLEDGEMENT=5 then

status reports will be sent on attempts 1, 6, 11, 16, *etc.* until the message is successfully delivered or returned. When it is delivered or returned, an acknowledgement is sent.

Set ACKNOWLEDGEMENT=0 if you only want acknowledgements sent when the message is successfully transmitted; set ACKNOWLEDGMENT=-1 if you want no acknowledgements or status reports sent.

See also ACK_SUCCESS.

BACKOFF (integer >= 0)
When BACKOFF specifies an integer greater than zero, backoff retries will be attempted for messages which could not be delivered immediately. See Section 26.4.1.8 for details. By default, BACKOFF=0.

BLOCK_ACK_TIMEOUT (integer > 0)
This option specifies how long to wait after transmitting a data block to the paging switch for an acknowledgement. The default value is 30 seconds. If no acknowledgement is received within BLOCK_ACK_TIMEOUT seconds after transmitting a block, then the page will be aborted and requeued for a subsequent delivery attempt. See the description of the BLOCK_TX_RETRIES option for further details on the use of this option.

BLOCK_TX_RETRIES (integer > 0)
After a data block is transmitted to the paging switch, the pager channel waits up to BLOCK_ACK_TIMEOUT seconds for an acknowledgement. If no acknowledgement is received within BLOCK_ACK_TIMEOUT seconds, then the page will be aborted and requeued for a subsequent delivery attempt. If a negative acknowledgement is received, then the pager channel will retransmit the data. Up to BLOCK_TX_RETRIES attempts will be made to send a data block. The page will be aborted and requeued for a subsequent delivery attempt when any one data block cannot be successfully transmitted after BLOCK_TX_RETRIES attempts (negative acknowledgements) or if the channel times out while waiting for an acknowledgement (no acknowledgement). The default value for BLOCK_TX_RETRIES is 7.

HEADER_STOP (character string)
This option specifies a character string to use as a delimiter between the message header lines and the message body. By default, the string “M:” is used.

ID_RX_RETRIES (integer > 0)
This option specifies the number of times the pager channel will attempt to obtain an “ID=” response from the paging switch. If an “ID=” is not received after ID_RX_TIMEOUT seconds, then the pager channel will transmit a carriage return to the switch and resume waiting for an “ID=” response. By default, the pager channel will try up to 4 times to obtain an “ID=” response. This default value can be changed with this option.

If an “ID=” response is not received after ID_RX_RETRIES attempts, then the page will be aborted and requeued for a subsequent delivery attempt.

ID_RX_TIMEOUT (integer > 0)
This option specifies the number of seconds to wait for an “ID=” response from the paging switch before timing out. The default value is 5 seconds. See the description of the ID_RX_RETRIES option for further details.

LINE_STOP (character string)
This option specifies a character string to use as a delimiter between each line of the filtered message. By default, a single space is inserted between each line.

LOGIN_RETRIES (integer > 0)
This option specifies the number of times the pager channel will attempt to log into a paging switch. After transmitting the password in response to an "ID=" received from the switch, the pager channel will wait up to LOGIN_ACK_TIMEOUT seconds for an acknowledgement. If no acknowledgement is received, then the pager channel will resume waiting again. If a negative acknowledgement or "ID=" response is received, then the password will be retransmitted and the channel will resume waiting for a response. If after going through this process LOGIN_RETRIES times and a successful login has not been established, the page will be aborted and requeued for a subsequent delivery attempt. In no case will the channel ever wait more than an accumulated LOGIN_ RETRIES * LOGIN_ACK_TIMEOUT seconds.

By default, a value of 4 is used for LOGIN_RETRIES.

LOGIN_ACK_TIMEOUT (integer > 0)
This option specifies the number of seconds to wait for a login acknowledgement from the paging switch before timing out. The default value is 5 seconds. See the description of the LOGIN_ACK_RETRIES option for further details.

LOGOUT_RETRIES (integer > 0)
This option specifies the number of times the pager channel will attempt to log off from a paging switch. After transmitting the logout command, the pager channel will wait up to LOGOUT_ACK_TIMEOUT seconds for an acknowledgement. If no acknowledgement is received, then the pager channel will merely close its connection to the paging switch and consider the pages to not have been sent.[9] If a negative acknowledgement is received, then the page was rejected. This is treated as a permanent error and the channel will return the message to the originator. If an unrecognized response is received, then the channel will resume waiting for a response. If, after waiting LOGOUT_RETRIES times only unrecognized responses have been received, the channel will merely close its connection and consider the pages to have been sent successfully. In no case will the channel ever wait more than an accumulated LOGOUT_RETRIES * LOGOUT_ACK_ TIMEOUT seconds.

By default, a value of 4 is used for LOGOUT_RETRIES.

LOGOUT_ACK_TIMEOUT (integer > 0)
This option specifies the number of seconds to wait for a logout acknowledgement from the paging switch before timing out. The default value is 5 seconds. See the description of the LOGOUT_ACK_RETRIES option for further details.

MAX_BLOCKS_PER_PAGE (integer)
Some paging switches impose a limit on the number of data blocks per page. The two typical cases are either no limit on the number of blocks per page (MAX_BLOCKS_PER_ PAGE=-1) or at most one block per page (MAX_BLOCKS_PER_PAGE=1). When a limit is imposed, each message will be broken into the requisite number of pages, no one page requiring more than MAX_BLOCKS_PER_PAGE data blocks to transmit.

No limit will be imposed when MAX_BLOCKS_PER_PAGE specifies a negative value. This is the default case, MAX_BLOCKS_PER_PAGE=-1.

This option and the MAX_PAGE_SIZE option are the two most important options used by the pager channel. They are also the most likely to vary from switch to switch. If the

[9] It's important to note that the logout response is the critical response from the paging switch which tells whether or not the pages were actually accepted. As such, the lack of a logout response cannot be treated as a success since to do so can result in lost pages.

proper values are not used, the paging switch will reject the pages presented to it.

MAX_DELIVERY_ATTEMPTS (integer > 0)
The MAX_DELIVERY_ATTEMPTS option controls the maximum number of delivery attempts to be made for a message. The default value is 15. If this option is set to 0, then no upper limit is imposed and PMDF's normal message return system will eventually return undeliverable messages. If a value is specified, then after delivery of the message has been tried MAX_DELIVERY_ATTEMPTS times, no further attempts will be made; the message will be immediately returned to the sender.

MAX_MESSAGE_SIZE (integer)
Messages which exceed, after filtering with the PAGER table mappings, a size of MAX_MESSAGE_SIZE bytes will be truncated to MAX_MESSAGE_SIZE bytes. The extra bytes are discarded. If MAX_MESSAGE_SIZE specifies a negative value, then no size limit will be imposed. This is the default setting, MAX_MESSAGE_SIZE=-1.

This option can be overridden on a per address basis with the MSGLEN address attribute.

MAX_PAGE_SIZE (integer)
Messages which exceed, after filtering with the PAGER table mappings, a size of MAX_PAGE_SIZE bytes will be split into multiple pages, no one page exceeding a byte count of MAX_PAGE_SIZE or requiring more than MAX_BLOCKS_PER_PAGE data blocks. If MAX_PAGE_SIZE specifies a negative value, then no size limit will be imposed. This is the default setting, MAX_PAGE_SIZE=-1.

This option can be overridden on a per address basis with the PAGLEN address attribute.

This option and the MAX_BLOCKS_PER_PAGE option are the two most critical options used by the pager channel. If the proper values are not used, the paging switch will reject the pages presented to it.

MAX_PAGES_PER_MESSAGE (integer)
Messages which require, after filtering with the PAGER table mappings, more pages than MAX_PAGES_PER_MESSAGE to transmit will be truncated to MAX_PAGES_PER_MESSAGE pages. The remaining pages are discarded. If MAX_PAGES_PER_MESSAGE specifies a negative value, then no pages will be discarded. This is the default setting, MAX_PAGES_PER_MESSAGE=-1.

This option can be overridden on a per address basis with the MAXPAG address attribute.

PAGES_PER_CALL (integer)
This option specifies the maximum number of pages to be sent per phone call (dialup session) to the paging switch. When a message is broken into more pages than PAGES_PER_CALL, then several calls will be placed so as to deliver all of the pages. In each call, no more than PAGES_PER_CALL pages will be sent. If PAGES_PER_CALL specifies a negative value, then no limit will be imposed on the number of pages which can be sent during any one call. This is the default setting, PAGES_PER_CALL=-1.

PASSWORD (character string)
Most, if not all, paging switches use the password "000000". If you happen to be using a switch which uses a different password, then that password must be specified with this option.

RETURN_HEADERS (0 or 1)

By default, the header lines for the originating message are appended to the bottom of acknowledgement and status reports generated by the pager channel. The display of these headers can be suppressed by specifying RETURN_HEADERS=0 in the option file. RETURN_HEADERS=1 yields the default behavior.

RETURN_PAGES (0 or 1)

By default, actual pages sent are not included in acknowledgement or status reports generated by the pager channel. To include the text of these pages, specify RETURN_PAGES=1. RETURN_PAGES=0 yields the default behavior.

REVERSE_ORDER (0 or 1)

When a message is broken into multiple pages, page 1/n, page 2/n, ..., page n/n, the pages are sent in reverse order: page n/n is sent first, followed by pages n-1/n, ..., 2/n, 1/n. This is done because many pagers present the most recently received page first (*i.e.,* some pagers store received pages in a last in, first out buffer). By specifying REVERSE_ORDER=0, multiple pages will be sent in the order page 1/n, 2/n, ..., n/n. The default setting is REVERSE_ORDER=1.

STRIP_8BIT_CHARS (0 or 1)

By default, characters with the eighth bit high are removed from the message. (They are removed prior to application of the PAGER mapping.) To preserve these characters,specify STRIP_8BIT_CHARS=0.

STRIP_CONTROL_CHARS (0 or 1)

By default, control characters (*i.e.,* characters with ordinal values 0 - 8, 10 - 31, and 127) are removed from the message. (They are removed prior to application of the PAGER mapping.) Each tab is converted to a single space prior to application of the PAGER mapping. To preserve these characters, specify STRIP_CONTROL_CHARS=0.

TEXT_CASE (LOWER, MIXED, NUMERIC, UPPER)

By default, the cases (upper case, lower case) of characters in a message are left undisturbed.[a] This default setting corresponds to TEXT_CASE=MIXED. By specifying either TEXT_CASE=LOWER or TEXT_CASE=UPPER, the message, after being filtered with the PAGER mapping table, will be converted entirely to lower or upper case characters. To only pass the numerals 0 through 9 to a pager, specify TEXT_CASE=NUMERIC.

USE_REPLY_TO (0 or 1)

One possible address to which acknowledgement and status report messages are sent is the original message's Reply-To: address. Under some circumstances it is never appropriate to use this address for this purpose. The USE_REPLY_TO option controls the use of the Reply-To: address for acknowledgements. A value of one (the default) causes the Reply-To: header to be used in the normal course of events. A value of zero inhibits use of the Reply-To: header for acknowledgements.

[a] Note that site supplied PAGER table entries can alter the case of characters in a message.

26.4.1.6 Use with PET Switches

When using a pager channel with a PET switch such as those used by Telecom Australia, the following options *must* be specified in the channel's option file

```
BLOCK_ACK_TIMEOUT=10
BLOCK_TX_RETRIES=5
LOGIN_RETRIES=2
LOGIN_ACK_TIMEOUT=10
LOGOUT_ACK_TIMEOUT=10
MAX_BLOCKS_PER_PAGE=1
MAX_PAGE_SIZE=156
PAGES_PER_CALL=6
PASSWORD=passwd
```

Failure to specify any or all of these options will render the pager channel incompliant with the PET specification (Telecom Australia document INB 028).

If you need to report problems to Telecom Australia, in so doing reference the University of Melbourne certification tests.

26.4.1.7 A Word on Determining Channel Options by Trial and Error

You can have to determine by trial and error the appropriate value for the MAX_PAGE_SIZE channel option.

To determine MAX_PAGE_SIZE, just try sending a long message (~500 bytes) and see what value you need for MAX_PAGE_SIZE to get the switch to accept the resulting pages. Be sure to enable master_debug for the channel so that you can see, blow by blow, the dialogue with the paging switch. *Be warned that the paging switch can respond by ignoring data sent to it and timeout* (*e.g.,* respond with a message like "Too Slow - Good Bye"). If this should happen, then decrease MAX_PAGE_SIZE and try again.

With some paging switches you can find that you need to pause a second or two and possibly even send a carriage return before issuing the go command in your modem script. (See, *e.g.,* Example 26–5.) With other switches, you can find that you cannot do this and must issue a go immediately after getting a CONNECT response from your modem. You'll have to experiment.

26.4.1.8 Frequent Delivery Retries with the BACKOFF Option

Normally, if a page cannot be delivered owing to a temporary error condition (*e.g.,* busy signal when dialing the paging switch), another delivery attempt will not be attempted until the next periodic delivery job runs as described in Section 1.4. However, for urgent pages, a retry in minutes rather than tens of minutes or hours is desirable. Rather than setting PMDF's periodic delivery job interval to a small value and then adjusting the service period for each channel with the `period` channel keyword, the BACKOFF mapping table and BACKOFF channel option can be used. With this mapping table and option, you can schedule exactly when to run jobs to attempt redelivery of undeliverable pages (*e.g.,* make attempts every minute for the first five minutes, then

every five minutes for the next thirty minutes, *etc.*). See Section 24.6 for complete details on the use of the BACKOFF mapping table and BACKOFF channel option.

26.4.1.9 Using a Directory Channel to Simplify Pager Addresses

A directory channel can be used to simplify pager addresses by creating a pseudodomain (*e.g.*, pager.example.com) and aliases (*e.g.,* user@pager.example.com) which translate to actual pager addresses (*e.g.,* /id=1234/paglen=100/@pactel.pager.example.com). These aliases insulate users from needing to know the actual pager addresses for individual pagers.

Directory channels are described in Section 3.2.

As an example, assume that the two channels pager_pactel and pager_mci and associated rewrite rules for the domains pactel.pager.example.com and mci.pager.example.com as shown in Section 26.4.1.2 have been added to the PMDF configuration file. Then a directory channel for the domain pager.example.com can be set up as follows:

1. Add to the PMDF configuration file the rewrite rule

```
pager.example.com          $U%pager.example.com@DIRECTORY-DAEMON
```

2. Add a directory channel to the configuration file:

```
directory
DIRECTORY-DAEMON
```

3. Create the world readable directory `pmdf_root:[directories]` (OpenVMS) or `/pmdf/directories` (UNIX) with the OpenVMS command

```
$ CREATE/DIR pmdf_root:[directories]/OWNER=[SYSTEM]
```

or the UNIX commands

```
# mkdir -mu=rwx,go= /pmdf/directories
# chown pmdf /pmdf/directories
```

On NT, the PMDF installation created this directory for you automatically.

4. In that directory create a directory database named `pager$example$com.dat` on OpenVMS or `pager.example.com` on UNIX with the PMDF CRDB (OpenVMS) or `pmdf crdb` (UNIX and NT) utility; *e.g.*, on OpenVMS

```
$ PMDF CRDB pager$example$com.txt pager$example$com.dat
```

or on UNIX

```
# pmdf crdb pager$example$com.txt pager.example.com
```

or on NT

```
C:\> pmdf crdb pager$example$com.txt pager.example.com
```

Here `pager$example$com.txt` is an input file with the format:

```
alias_1      value_1
alias_2      value_2
.            .
.            .
.            .
```

with *alias_1*, *alias_2*, ... the names of aliases and *value_1*, *value_2*, ... the associated translation values. A sample `pager$example$com.txt` file is shown in Example 26–7.

Example 26–7 Sample `pager$example$com.txt` File

```
andy         /id=1234/paglen=100/@pactel.pager.example.com
sue          /id=1122/paglen=100/@pactel.pager.example.com
joe          /id=4321/paglen=300/@mci.pager.example.com
spare        /id=1111/paglen=100/@mci.pager.example.com
support      /id=0101/@pactel.pager.example.com
```

Of course, steps 2 and 3 should be omitted if a directory channel has previously been set up.

26.4.2 Pager Channel Addresses

A pager channel address can take one of two forms:

1. *pager-id*@*domain-name* where *pager-id* is the pager identification number for the pager to which a message is to be sent, and *domain-name* is one of the pager domain names added to the configuration file as described in Section 26.4.1.2. An example of such an address is 1234@pager.example.com.
2. *AVPL*@*domain-name* where *AVPL* item, an attribute-value pair list, is described in the following subsections. An example of such an address is /ID=1234/@pager.example.com.

26.4.2.1 The Contents of the Attribute-value Pair List (AVPL)

The left hand side of a pager channel address in AVPL form, (*i.e.,* the *AVPL* in *AVPL*@*domain-name*, is constructed in the linear attribute-value pair list (AVPL) format recommended by RFC 987 and RFC 1148. The general form is:

/attribute1=value1/attribute2=value2/.../

The attribute codes specify the destination pager number and its characteristics. The available attributes are shown in Table 26–3.

Table 26–3 Pager Channel Addressing Attributes

Attribute name	Usage
ID	Pager ID number.
MAXPAG	Maximum number of pages to send for any one message. If a message is broken into more than MAXPAG pages, then pages MAXPAG, MAXPAG+1, MAXPAG+2... are discarded and not sent.
MSGLEN	Maximum length in bytes of the message to send as one or more pages. Message will be truncated to this length if necessary. This will override any size set with the MAX_MESSAGE_SIZE channel option.
PAGLEN	Maximum length in bytes per page; message will be broken into multiple pages with no one page exceeding this length. This will override any size set with the MAX_PAGE_SIZE channel option.

When an AVPL oriented pager address is used, the ID attribute must be specified.

If any of the characters "/", "=", and "$" are to appear within values in the AVPL, then they must be quoted. See Section 26.6.2.2 for instructions on how to quote these characters.

26.4.2.2 Examples of Pager Channel Addresses

Some sample pager addresses are shown below:

```
1234@mci.pager.example.com
/id=1234/@mci.pager.example.com
/paglen=100/id=1234/@mci.pager.example.com
/msglen=400/id=1234/@mci.pager.example.com
/maxpag=2/id=1234/@mci.pager.example.com
/id=1234/paglen=200/msglen=400/@mci.pager.example.com
```

26.4.3 Pager Channel Logging

The `logging` channel keyword can be used with pager channels to enable logging activity. Activity is logged in the PMDF mail log file. See Section 2.3.4.84 for a description of this file. The "type of entry" item in each log file will be a single character selected from the set A, F, R, S:

Logging Code	Error Type	Description
A	Permanent	Message aborted; maximum delivery attempts exceeded
F	Temporary	Error occurred while communicating with the paging switch
R	Permanent	Message rejected by the paging switch
S	Success	Message sent successfully

In the case of a temporary error, the message is requeued for a subsequent delivery attempt provided that the maximum number of delivery attempts has not been exceeded.

If the maximum number of delivery attempts has been exceeded, then the error is promoted to a permanent error. In the case of a permanent error, the message is returned to the message's originator.

26.4.4 Identifying Troublesome Modems

When the pager channel finds one or more modems to be unusable, it will note the device names.

On OpenVMS, these device names are sent via an OPCOM broadcast to all operator classes, or if the site-supplied command procedure `PMDF_COM:bad_modem_alert.com` exists, that command procedure is instead executed. This site-supplied command procedure `bad_modem_alert.com` can obtain the list of bad modems from the DCL symbol PMDF_BAD_MODEMS. The value of that symbol is a comma separated list of the bad device names. The name of the currently running channel can be obtained from the OpenVMS logical PMDF_CHANNEL.

On UNIX, these device names are written to a file `/tmp/pmdf_bad_modem-`*`uniqueid`*, where *`uniqueid`* is a unique string generated to disambiguify the file name. In addition, if a site-supplied `/pmdf/bin/bad_modem_alert` shell script exists, it will be executed, and the bad modem device names will be supplied to it via standard input, `stdin`.

On NT, these device names are written to a file `C:\tmp\pmdf_bad_modem-`*`uniqueid`*, where *`uniqueid`* is a unique string generated to disambiguify the file name. In addition, if a site-supplied `C:\pmdf\bin/bad_modem_alert` shell script exists, it will be executed, and the bad modem device names will be supplied to it via standard input, On UNIX, these device names are written to a file `/tmp/pmdf_bad_modem-`*`uniqueid`*, where *`uniqueid`* is a unique string generated to disambiguify the file name. In addition, if a site-supplied `/pmdf/bin/bad_modem_alert` shell script exists, it will be executed, and the bad modem device names will be supplied to it via standard input, `stdin`.

26.5 Pipe Channels (OpenVMS and UNIX)

Pipe channels are used to effect delivery for specific addresses via a site-supplied program or script. While pipe channels are loosely based upon the | (pipe) functionality of sendmail, they have been carefully designed to not pose a security threat. First, you, as the postmaster, have to manually add a pipe channel to your configuration before it can be used. If you do not trust the technology, then do not add one to your configuration. Second, the commands executed by the pipe channel are controlled by you, as the postmaster, and no user supplied input can find its way into those commands: each command you supply is used verbatim with the exception of the optional substitution of a filename generated by the channel itself without reference to user supplied input. Finally, the decision to run a command is not based upon the presence of special characters appearing in a recipient address, but rather upon a recipient address exactly matching a specific address or host name in a table or database which you must provide. If an exact match between an incoming address and your table or database exists, then your command listed in the table for that match is executed.

Unlike the sendmail pipe functionality, the PMDF pipe channel does not pipe the message to be processed to the program or script. Instead, it writes the message to be processed to a temporary file and then forks a subprocess to run the site-supplied command for that message. That command should make use of the name of the temporary file which can be substituted into the command by the channel. The temporary file should not be deleted or altered by the subprocess; the channel will delete it itself. If it is not possible to prevent the subprocess from disrupting the file, then the pipe channel should be marked with the `single` channel keyword.

VMS

On OpenVMS systems, if the subprocess exits with a successful completion code (*i.e.,* an odd valued completion code), then the message is presumed to have been delivered successfully. If it exits with a completion code of PMDF__NO (decimal value 178028690, hexadecimal value 0A9C8092), then the message is returned as undeliverable. Delivery of the message will be deferred if any other completion code is returned.

UNIX

On UNIX systems, if the subprocess exits with exit code of 0 (EX_OK) then the message is presumed to have been delivered successfully and is removed from PMDF's queues. If it exits with an exit code of 71, 74, 75, or 79 (EX_OSERR, EX_IOERR, EX_TEMPFAIL, or EX_DB) then a temporary error is presumed to have occurred and delivery of the message is deferred. If any other exit code is returned, then the message will be returned to its originator as undeliverable. These exit codes are defined in the system header file `sysexits.h`.

26.5.1 Setting Up the Channel

There are two steps in setting up a pipe channel: (1) adding the channel to the PMDF configuration file, and (2) setting up the channel option file, pipe database or (UNIX only) profile database entries to specify particular commands for particular user or host addresses.

26.5.1.1 Adding the Channel to the Configuration File

Note: While you can configure multiple pipe channels, in general you only need to configure a single pipe channel.

A message to be processed by a pipe channel is usually routed to the channel via a combination of an alias and rewrite rules. For instance, the system example.com might want all mail for the addresses info-pmdf@example.com and gripes@example.com to be routed to a pipe channel. This could be accomplished with the alias file entries

```
info-pmdf: info-pmdf@pipe.example.com
gripes: gripes@pipe.example.com
```

where `pipe.example.com` is in turn a host name associated with a pipe channel via rewrite rules. For instance,

```
pipe.example.com          $u%pipe.example.com@PIPE-DAEMON
```

So, to configure a pipe you need to determine the host names, pipe1.domain, pipe2.domain, ... which you want to use. Once you have determined these, add them to the rewrite rules section of your PMDF configuration file:

```
pipe1.domain            $u%pipe1.domain@PIPE-DAEMON
pipe2.domain            $u%pipe2.domain@PIPE-DAEMON
...                     ...
```

Then, to the end of your PMDF configuration, add the definition of the pipe channel itself:

```
pipe
PIPE-DAEMON
```

Be sure to include a blank line *before and after* this channel definition.

On UNIX, the pipe channel normally runs as user pmdf. So on UNIX, if you want the pipe channel to run as some other user, you can use the user channel keyword to specify the desired username. Note that the argument to user is normally forced to lowercase, but original case will be preserved if the argument is quoted.

At this point, the pipe channel has been added to the configuration. However, it cannot be used until you create a channel option file, or a pipe database, or (UNIX only) define and set delivery methods for pipe channel addressees, as described next.

26.5.1.2 Profile Database Entries for Pipe Channel Addressees (UNIX only)

On UNIX, before looking in the channel database or option file, a pipe channel first queries the PMDF profile database checking whether there is a delivery method set for the addressee. Only if there is no such entry is the pipe database or pipe option file consulted.

A PMDF profile database delivery method entry for a pipe channel addressee is similar to that for a local (L) channel addressee, except that the pipe channel domain is used. For instance,

```
# pmdf profile
profile> set delivery mailworks -user=jane.doe@pipe.example.com
```

26.5.1.3 Pipe Database

Rather than placing user or host addresses and corresponding commands in a pipe channel option file, described below in Section 26.5.1.4, or on UNIX using profile database entries for users, described above in Section 26.5.1.2, user or host entries can be placed in the PMDF pipe database. The pipe database can be particularly useful for sites with large numbers of entries (more efficiently stored in a database than in an option file), or for sites where the user addresses and command strings are rather long (in which case a "long" database created with the /LONG_RECORDS (OpenVMS) or -long_records qualifier (UNIX) can be used).

The pipe database is referenced via the PMDF_PIPE_DATABASE logical (OpenVMS) or PMDF tailor file option (UNIX), hence it is usually PMDF_TABLE:pipe.dat on OpenVMS or /pmdf/table/pipedb.* on UNIX. This database is a regular PMDF CRDB (OpenVMS) or pmdf crdb (UNIX) database created from a text input file.

The format of entries in the input text file should be:

```
address1     command1
address2     command2
...                          ...
```

where the addresses, *address1*, *address2*, ..., can be either of the form *user@host* or *host*. PMDF probes first for user-specific entries and only if no user-specific entry is found, will PMDF then look for a host entry. See Section 26.5.2 below for an additional discussion of the order in which probes of various forms are made to the various possible entry sources.

On OpenVMS, such an input text file would be turned into a pipe database using the commands:

```
$ PMDF CRDB input-file-spec pipe.tmp
$ RENAME pipe.tmp PMDF_PIPE_DATABASE
```

On UNIX, use the commands:

```
# pmdf crdb input-file-spec PMDF_PIPE_DATABASE
```

26.5.1.4 Option Files

Unless you have a pipe database, or have established delivery methods for pipe channel addressees, each pipe channel must have an option file. If there are no delivery methods set in the PMDF profile database, and no option file nor pipe database, then the channel will not operate.

The commands to execute for each envelope recipient address presented to the channel are specified in the PMDF profile database, in the pipe database or in the pipe channel's option file. If an address does not appear in one of these locations, then an error notification is sent back to the message originator.

Pipe channel option files are stored in the PMDF table directory and have names of the form *x*_option, where *x* is the name of the pipe channel to which the option file applies. (In most instances, the file name will be pipe_option; *i.e.*, PMDF_TABLE:pipe_option. on OpenVMS or /pmdf/table/pipe_option on UNIX.)

To process the address *user@host*, the pipe channel first probes the option file for an entry of the form

```
user@host=command
```

If no matching entry is found, the channel next probes the option file for an entry of the form

```
host=command
```

If still no matching entry is found, then the recipient address is deemed bad and an error notification is sent back to the message originator. See Section 26.5.2 below for an additional discussion of the order in which probes of various forms are made to the various possible entry sources.

If, however, a probe does find a matching entry, then the specified command, *command*, is executed. Prior to being executed, any occurrences of the phrase `%s` appearing in *command* are replaced with the name of the temporary file containing the message to be processed. It is important that the command to be executed neither delete nor otherwise alter the temporary message file as it can be needed for further pipe channel recipients of the same message. If disruption of the message file cannot be prevented, then mark the channel with the `single` channel keyword.

The command to be executed will be run by a subprocess of the process running the pipe channel. As such, it will be running with the privileges of the PMDF processing account (usually the SYSTEM account on OpenVMS or `pmdf` account on UNIX). See Section 26.5 for a description of the exit or completion codes with which the command should exit the subprocess.

Note: As with any PMDF option file, it is important that the option file not be world writable. This is especially true of pipe channel option files.

In addition to the command entries in the pipe channel option file, there is one additional general option available:

SHELL_TIMEOUT (integer; UNIX only)
The SHELL_TIMEOUT option can be used to control how long in seconds the channel will wait for a shell command to complete. Upon such time outs, the message will be returned back to the original sender with an error message along the lines of "Timeout waiting for ...'s shell command ... to complete". The default value is 600 (corresponding to 10 minutes).

26.5.2 The Order in Which Entries are Checked

The logic for checking entries in the profile database (UNIX only), pipe database and pipe channel option file is as follows:

1. (UNIX only) Check the profile database for a *user@host* entry.
2. Check the database for a *user@host* entry.
3. Check the option file for a *user@host* entry.
4. Check the database for a *host* entry.
5. Check the option file for a *host* entry.

If no profile database entry exists (as is always the case on OpenVMS) and if the pipe database does not exist, then only the pipe option file is checked, *i.e.*, steps (3) and (5) only. If no profile database entry exists (as is always the case on OpenVMS) and if the database file exists but cannot be opened, then the message is passed over — the condition is treated as a temporary error. And when checks (1)–(5) turn up no results, the message is bounced.

26.5.3 Example Usage

Suppose that messages for the two addresses `wombat@example.com` and `sflovers@example.com` are to be processed by a site-supplied programs using the command

```
/usr/local/uurec < filename
```

where *filename* is the name of the input file to process.

Suppose further that example.com is the official local host name for the site so that entries for wombat and sflovers in the alias file will serve to redirect messages to those two addresses. Then, the entries

```
wombat: wombat@pipe.example.com
sflovers: sflovers@pipe.example.com
```

should be added to the alias file. Next, add the following entries to the pipe channel option file:

```
wombat@pipe.example.com=/usr/local/uurec < %s
sflovers@pipe.example.com=/usr/local/uurec < %s
```

With this configuration, messages for sflovers@example.com are rerouted to the pipe channel using the address sflovers@pipe.example.com. The pipe channel, upon receipt of the message, will then execute the supplied command to process the message.

26.6 Printer Channels

Printer channels can be used to route e-mail to a spooled printer queue. While the current implementation of the printer channel is designed for use with "dumb" printers, it is flexible enough to do sophisticated printing on "intelligent" printers such as PostScript printers as described in *e.g.,* Section 26.6.4.

The addressing format required to direct e-mail to a printer channel is presented in Section 26.6.2.

On OpenVMS systems, the printer channel interfaces to printer queues via the $SNDJBC system service. On UNIX systems, the channel forks a child to print each message file. The command used to issue the print request can be controlled with the PRINT_COMMAND option described in Section 26.6.1.2. By default, an `lpr` command is used.

VMS

On OpenVMS, **do not ever specify an execution queue** for a printer channel queue; (*i.e.*, make sure that the printer channel queue is an **output queue**). Picture the following scenario: a printer channel is marked `headeromit` (or `headerbottom`) and a message containing only a command procedure is sent to the printer channel for "printing". This message — a command procedure — will then be submitted to the

execution queue under the guise of the user running the printer channel, who is typically a privileged user.

26.6.1 Setting Up the Channel

There are two steps in setting up a printer channel: (1) adding the channel to the PMDF configuration file, and (2) setting up a channel option file, if necessary. For most "dumb" printers (*e.g.,* a line printer), an option file is not required. In addition, no option file is required of "intelligent" printers (*e.g.,* PostScript printers), if the printer's queue is managed by a symbiont which knows how to deal with raw ASCII text files.

VMS

If you want to select the usernames under which print jobs should be submitted, then specify SET_USERNAME=1 in the channel option file. The USERNAME addressing attribute can then be used to select the username under which a given message should be submitted. This makes it possible for banner and trailer pages to bear the name of the intended recipient rather than the username of the process running the printer channel.

26.6.1.1 Adding the Channel to the Configuration File

First determine the names of the printer queues (OpenVMS) or print job destinations (UNIX) to which you will be directing mail and then select a domain name to identify with each queue. These domain names are used to route mail to a particular queue. For instance, suppose mail is to be routed to the printer SYS$PRINT on the host vaxa.example.com. Then an appropriate domain name might be sysprint.vaxa.example.com.

Once the printer names are determined and domain names selected, add a channel block of the form

```
printer single 733
PRINTER-DAEMON
domain-name-1    printer-name-1
domain-name-2    printer-name-2
.
.
.
```

to the PMDF configuration file. Here *domain-name-1*, *domain-name-2*, ... and *printer-name-1*, *printer-name-2*, ... are, respectively, the selected domain name and the printer names. On OpenVMS you must use printer queue names. On UNIX, you must use printer names suitable for use in an `lpr` command with the `-P` switch.

Continuing with the SYS$PRINT and vaxa.example.com example from above, the channel block would be

```
printer single 733
PRINTER-DAEMON
sysprint.vaxa.example.com   SYS$PRINT
```

After adding the channel block, go to the top of the configuration file and add rewrite rules of the form shown below:

```
domain-name-1          $U@domain-name-1
domain-name-2          $U@domain-name-2
.
.
.
```

For instance, in the context of our example, the following rewrite rules can be added

```
sysprint                     $U@sysprint.vaxa.example.com
sysprint.vaxa.example.com    $U@sysprint.vaxa.example.com
```

The first rewrite rule shown is not necessary: it merely allows PMDF to recognize the address *avpl*@sysprint as a short form address for *avpl*@sysprint.vaxa.example.com.

Note: If you are part of a TCP/IP network, then you can want to add the printer domain names you selected to your DNS using MX records. This will allow other machines to route mail to your printer queues.

26.6.1.2 Option Files

In order to properly print messages on the printers associated with the channel, it can be necessary to use an option file. With an option file, the following items can be specified:

- introductory text or commands which must be sent to the printer — referred to below as a "preamble";
- commands which must proceed and/or follow lines of text to be printed;
- commands to eject a page and/or terminate the print job; and
- various options to be passed to the print symbiont.

Option files are used to set run-time options on a per channel basis. Option files are stored in the PMDF table directory and have names of the form *x*_option, where *x* is the name of the printer channel to which the option file applies. (In most instances, the file name will be printer_option; *i.e.*, PMDF_TABLE:printer_option. on OpenVMS or /pmdf/table/printer_option on UNIX.)

Note that an option file applies to an entire printer channel; a printer channel which can be serving more than one printer queue. If you have printer queues which require different option files, then you should set up multiple printer channels (giving the channels different names such as printer1 and printer2 instead of printer).

Option files consist of several lines. Each line contains the setting for one option. An option setting has the form:

```
option=value
```

value can be either a string or an integer, depending on the option's requirements. If the option accepts an integer value, *value*, a base can be specified using notation of the form *b*%*v*, where *b* is the base expressed in base 10 and *v* is the actual value expressed in base *b*.

Comments are allowed. Any line that begins with an exclamation point, !, is considered to be a comment and is ignored. Blank lines are also ignored in any option file.

The available options are:

AT (string <= 252 characters long)
This option can be used to specify the content of the text string printed along with the contents of any /AT=*value* attribute-value pair (AVP) appearing in the To: address presented to the printer channel. If no AT option is specified in the option file, then the default string "Attention: " will be used. For example, if the To: address passed to the printer channel is

```
"/AT=Mrocheck Doe/MS=DH X.25/"@printer.example.com
```

then the text

```
Attention:        Mrocheck Doe
```

will be printed on the first page of the message.

BURST (0 or 1; OpenVMS only)
The BURST option controls whether two file flag pages with a burst bar between them are printed preceding the mail message. BURST=0 requests that no burst pages be printed; BURST=1 requests that the burst pages be printed preceding the message. By default, no burst pages are requested (BURST=0).

END_ATTRIBUTE (string <= 252 characters long)
The END_ATTRIBUTE option specifies a text string to be printed immediately following the display of an attribute value extracted from the mail message's To: address. By default, no text string is printed after displaying an attribute value.

END_COVER (string <= 252 characters long)
The END_COVER option specifies a text string to be printed at the end of the cover page printed prior to the message body. By default, a form feed character is printed at the end of the cover page.

END_HEADERLINE (string <= 252 characters long)
The END_HEADERLINE option specifies a text string to be printed immediately following a header line from the mail message (*e.g.,* To:, From:, Subject:, *etc.*). By default, no text string is printed after a header line.

END_JOB (string <= 252 characters long)
The END_JOB option specifies a text string to be printed at the end of the mail message. By default, no text string is printed.

END_LINE (string <= 252 characters long)
The END_LINE option specifies a text string to be printed immediately following a line from the mail message's body. By default, no text string is printed after a message body line.

FLAG (0 or 1; OpenVMS only)
The FLAG option controls whether a flag page is printed preceding the mail message. FLAG=0 requests that no flag page be printed; FLAG=1 requests that a flag page be printed preceding the message. By default, no flag page is requested (FLAG=0). The setting of this option has no effect if BURST=1 is specified in the option file.

FORM (string <= 252 characters long; OpenVMS only)
Specifies the name or number of the print form to be associated with the message when it is submitted to the printer queue. By default, no print form is associated with messages when they are submitted for printing.

HEADER_OPTIONS (filename <= 252 characters long)
This option specifies the name of a header option file to use when parsing message header lines. By default, no header option file is used. If the channel is tagged `headeromit` in the channel block in the PMDF configuration file, this option is ignored.

MIME_HANDLING (0 or 1)
By default, the printer channel will interpret the MIME structure of MIME messages, decoding and printing each part of a multipart message. This corresponds to specifying MIME_HANDLING=1. To supress the MIME interpretation of messages and have their content just printed as an ordinary RFC 822 message, specify MIME_HANDLING=0.

MS (string <= 252 characters long)
This option can be used to specify the content of the text string printed along with the contents of any /MS=*value* attribute-value pair (AVP) appearing in the To: address presented to the printer channel. If no MS option is specified in the option file, then the default string "Mail stop: " will be used. For example, if the To: address passed to the printer channel is

```
"/AT=Mrocheck Doe/MS=DH X.25/"@printer.example.com
```

then the text

```
Mail stop:        DH X.25
```

will be printed on the first page of the message.

NOTIFY (0 or 1; OpenVMS only)
Generate a notification broadcast when a mail message is printed. By default, this notification is not generated, NOTIFY=0. It will only be generated when NOTIFY=1, SET_USERNAME=1, and a username has been specified in the printer address. In that case, the specified user will receive a broadcast message after their mail message has completed printing.

O (string <= 252 characters long)
This option can be used to specify the content of the text string printed along with the contents of any /O=*value* attribute-value pair (AVP) appearing in the To: address presented to the printer channel. If no O option is specified in the option file, then the default blank string will be used.

OU (string <= 252 characters long)
This option can be used to specify the content of the text string printed along with the contents of any /OU=*value* attribute-value pair (AVP) appearing in the To: address presented to the printer channel. If no OU option is specified in the option file, then the default blank string will be used.

P1–P8 (string <= 252 characters long; OpenVMS only)
These options can be used to specify the content of the text string printed along with the contents of any /P1=*value*, ..., /P8=*value* attribute-value pair (AVP) appearing in the To: address presented to the printer channel. If no P1, P2, ..., or P8 option is specified in the option file, then the default blank string will be used.

P1_DEFAULT–P8_DEFAULT (string <= 252 characters long; OpenVMS only)

These eight options set values for the parameters P1 through P8 to be specified when submitting a print job. Unless a message specifically sets any of P1 through P8 with addressing attributes, these default values will be used. If a message specifies any of P1 through P8, then none of these default values will be used and the values specified by the message will instead be used.

PAGINATE (0 or 1; OpenVMS only)

Specify whether or not the printer symbiont should paginate the output by inserting a form feed whenever output reaches the bottom margin of the output form. PAGINATE=0 disables the insertion of form feeds; PAGINATE=1 enables the insertion of form feeds. By default, form feeds are inserted by the symbiont (PAGINATE=1).

PREAMBLE (string <= 252 characters long)

The PREAMBLE option specifies a text string to print at the top of the cover page for the message. If the first character in the string is an at-sign, @, then the remainder of the string is interpreted as a file name and the contents of that file is first sent to the printer prior to the cover page. (This mechanism can be used to download code to intelligent printers such as PostScript printers.)

PRINT_COMMAND (string <= 252 characters long; UNIX only)

Specifies the format of the command to issue to print a message. By default, the `lpr` command is used with the format

```
PRINT_COMMAND=lpr -P%p -r %f
```

See Section 26.6.1.2 for further details.

QUOTE_CHARS (string <= 252 characters long; UNIX only)

Specifies the shell metacharacters which require quoting with a backslash, \, in order to literalize them. By default,

```
QUOTE_CHARS="#$&'()*;<=>?[\]`{|}
```

is used. Specify

```
QUOTE_CHARS=
```

to disable the quoting of characters. See Section 26.6.1.3 for further details.

SET_USERNAME (0 or 1; OpenVMS only)

USERNAME addressing attributes-value pairs are ignored by default and not passed on to the print subsystem when the message is spooled for printing. By specifying SET_USERNAME=1 in the option file, these attribute-value pairs will be honored; the spooled print job will be submitted using the specified username. Process running the channel — typically SYSTEM — must have CMKRNL privilege in order to use this option. The channel itself does not make use of this privilege: the $SNDJBC system service requires it of any process which submits jobs under a different username.

The username under which PMDF channel programs run usually has CMKNRL privilege. This username is given by the PMDF_BATCH_USERNAME logical.

SETUP (string <= 252 characters long)

This option can be used to specify the content of the text string printed along with the contents of any /SETUP=*value* attribute-value pair (AVP) appearing in the To: address

presented to the printer channel. If no SETUP option is specified in the option file, then the default blank string will be used.

SETUP_DEFAULT (string <= 252 characters long; OpenVMS only)
This option allows specification of a setup module to be used by the print symbiont. This setup module will be specified unless the printer address itself specifies a setup module. In that case, the module specified in the address is instead used. To specify multiple module names, specify each name separated by commas; *e.g.*,

```
SETUP_DEFAULT=module1,module2,module3
```

START_ATTRIBUTE (string <= 252 characters long)
The START_ATTRIBUTE option specifies a text string to be printed immediately prior to the display of an attribute value extracted from the mail message's To: address. By default, no text string is printed before displaying an attribute value.

START_HEADERLINE (string <= 252 characters long)
The START_HEADERLINE option specifies a text string to be printed immediately before a header line from the mail message (*e.g.,* To:, From:, Subject:, *etc.*). By default, no text string is printed before a header line.

START_LINE (string <= 252 characters long)
The START_LINE option specifies a text string to be printed immediately before a line from the mail message's body. By default, no text string is printed before a message body line.

TN (string <= 252 characters long)
This option can be used to specify the content of the text string printed along with the contents of any /TN=*value* attribute-value pair (AVP) appearing in the To: address presented to the printer channel. If no TN option is specified in the option file, then the default string "Telephone number: " will be used. For example, if the To: address passed to the printer channel is

```
"/AT=Mrocheck Doe/TN=900-555-1212/"@printer.example.com
```

then the text

```
Telephone number: 900-555-1212
```

will be printed on the first page of the message.

TRAILER (0 or 1; OpenVMS only)
The TRAILER option controls whether a file trailer page is requested when the message is submitted to the printer queue. TRAILER=0 requests that no trailer page be printed; TRAILER=1 requests that a trailer page be printed. By default, no trailer page is requested (TRAILER=0).

26.6.1.3 Controlling the Print Command (UNIX)

On UNIX systems, the print command issued by the printer channel can be controlled with the PRINT_COMMAND channel option. With that option, the format of the print command to issue is specified:

```
PRINT_COMMAND=command
```

where *command* is the command to issue. command can contain two special sequences, %f and %p. Should they appear, they will be replaced, respectively, with the name of the file to print and the name of the printer to use. The PRINT_COMMAND should be such as to delete the file after printing. The printer name is the name given in the channel definition (*e.g., printer-name-1*, *printer-name-2*, ... in Section 26.6.1.1). To specify a literal % or \ in the print command, use \% or \\ in *command*.

For example, the default value of the PRINT_COMMAND option is

```
PRINT_COMMAND=lpr -P%p -r %f
```

Thus with this default, when given a printer name of PS_TIPSY and file name of /pmdf/queue/printer/spool/ZZAX02G.00, the resulting print command would be

```
lpr -PPS_TIPSY -r /pmdf/queue/printer/spool/ZZAX02G.00
```

Note that the -r switch of the lpr command requests that the file be deleted after it has been printed.

Prior to be being substituted into the print command, any occurrences of the characters

```
" # $ & ' ( ) * ; < = > ? [ \ ] ` { | }
```

in the printer name or file name are preceded with a backslash, \, so as to literalize them and prevent them from being interpreted as shell metacharacters. Use the QUOTE_CHARS channel option to change the list of characters requiring quoting. To inhibit the quoting of characters, specify a null string with the QUOTE_CHARS option.

Utmost care is taken to ensure that user-supplied information is not injected into the command line executed by the forked child. As such, several of the features found in the OpenVMS printer channel are not available under UNIX (*e.g.,* the ability to specify the username under which to print, use of the destination address in the print job name, specification of printing options, *etc.*).

26.6.1.4 Handling Multipart Messages

By default, the printer channel will interpret MIME messages, decoding and printing each part of the message. This is likely to cause problems when a user sends a binary message part such as an executable to a printer channel. There are two ways to deal with this: (1) disable entirely the interpretation of MIME messages, printing the entire message as a single RFC 822 message without decoding any encoded message parts, or (2) using a conversion channel to discard unwanted message parts.

While (1) is easily implemented by just specifying MIME_HANDLING=0 in the channel option file, it can result in printing large amounts of unwanted data (*e.g.,* printing an encoded executable). Approach (2) is therefore more practical. See Section 22.1 for complete information on configuring a conversion channel. Note that when using this approach, it's best to use entries in the conversions file which explicitly accept the message content types which you will allow to be printed and then discard all other types.

For instance, to print only message parts of type text and substitute for all other message parts a text string stating that an original part was discarded, on OpenVMS you might use:

```
out-chan=printer; in-type=text; in-subtype=*;
  command="COPY 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=printer; in-type=*; in-subtype=*;
  out-type=text; out-subtype=plain; out-mode=text; out-encoding=none;
  command="@PMDF_TABLE:discard.com"
```

where `PMDF_TABLE:discard.com` is the site-supplied DCL command procedure

```
$ SIZE = F$FILE_ATTRIBUTES (INPUT_FILE, "EOF")
$ OPEN/WRITE FILE 'OUTPUT_FILE'
$ WRITE FILE -
  "['''SIZE' block ''INPUT_TYPE'/''INPUT_SUBTYPE' message part discarded]"
$ CLOSE FILE
```

Similarly, on UNIX you might use:

```
out-chan=printer; in-type=text; in-subtype=*;
  command="COPY 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=printer; in-type=*; in-subtype=*;
  out-type=text; out-subtype=plain; out-mode=text; out-encoding=none;
  command="/pmdf/table/discard.sh"
```

where `/pmdf/table/discard.sh` is the site-supplied shell script

```
size=`wc -c $INPUT_FILE | awk '{print $1}'`
echo '['$size 'byte' $INPUT_TYPE'/'$INPUT_SUBTYPE \
 'message part omitted]' > $OUTPUT_FILE
```

Alternatively, to simply discard non-text parts silently, without bothering to include a text note explaining that a part was discarded, you could use DELETE=1 in the conversion entry, *e.g.*,

```
out-chan=printer; in-type=text; in-subtype=*;
  command="COPY 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=printer; in-type=*; in-subtype=*;
  delete=1
```

26.6.2 Printer Channel Addresses

Printer channel addresses are straightforward but do take a few minutes to learn. PMDF-FAX users will find printer addresses quite familiar.

The general format of an address for a printer channel is:

```
AVPL@printer-domain-name
```

where *printer-domain-name* is one of the printer domain names added to the configuration file as described in Section 26.6.1.1. The *AVPL* item is described in following subsections.

26.6.2.1 The Contents of the Attribute-value Pair List, AVPL

The left hand side of a printer channel address, (*i.e.,* the *AVPL* in *AVPL@printer-domain-name*, is constructed in the linear attribute-value pair list (AVPL) format recommended by RFC 2156. The general form is:

```
/attribute1=value1/attribute2=value2/.../
```

The trailing / at the end of the AVPL is here superfluous but nonetheless called for by the RFCs. PMDF does not require that it be supplied; however, future releases of PMDF can require its presence so it is best not to get into the habit of omitting it.

The attribute codes, for the most part, describe the intended recipient of the printer output. The available codes are similar to X.400 attribute codes, but they are not identical. They must be chosen from those shown in Table 26–4. Note that the P1 — P8, SETUP, and USERNAME attributes are only supported on OpenVMS.

Table 26–4 Printer Channel Addressing Attributes

Attribute name	Usage
AT	Attention (usually a person's name)
MS	Mail stop
O	Organization or company
OU	Organizational unit or department
P1 — P8	Parameters to be passed through the print job; OpenVMS only
SETUP	Setup module to be used by the print job; specify multiple modules in a comma separated list; OpenVMS only
TN	Recipient's telephone number
USERNAME	Username under which to submit print job; OpenVMS only

There are no mandatory attributes; however, the use of at least AT is recommended so that the printer operator can identify for whom the printer output is intended. Attributes can be specified in any order and can be specified multiple times. If more than one P1 attribute is specified, then only the rightmost one will be used. The same holds for the other parameter attributes, P2 through P8.

USERNAME attributes will be ignored unless the SET_USERNAME=1 option has been specified in the channel's option file. Use of this attribute allows specification of the username under which to submit each print job. This username, rather than that of the process running the printer channel, will then appear on any banner or trailer pages.

26.6.2.2 Quoting the AVPL

The attribute-value pair list (AVPL) can contain spaces and other special characters. If so, the entire list must be enclosed in double or single quotes.

The characters /, =, and $ are treated as list punctuation characters. If any of these characters appear in either attribute names or values, they must be prefixed with a dollar sign, $, to remove their special meaning. For instance, the "a/s" in the name "Example a/s" requires special quoting:

```
IN%"'/at=Rex Sheds/O=Example a$/s/'@printer.example.com"
```

26.6.2.3 Examples of Printer Channel Addresses

Assume that the domain name printer.example.com is a legal printer domain name. Then, the following are examples of legal printer channel addresses as they would be specified to either VMS MAIL or DECwindows MAIL:

```
IN%"'/AT=Ralph/TN=(714) 624-7907/'@printer.example.com"
IN%"'/TN=(714) 621-8465/AT=John Doe/O=Example/'@printer.example.com"
IN%"'/AT=Dan/O=Example/OU=MIS Dept./MS=XJ 614/'@printer.example.com"
IN%"'/AT=Dan Doe/USERNAME=DAN/'@printer.example.com"
```

When using a user agent which accepts plain RFC 822 addresses (*e.g.,* PMDF MAIL or Pine), use instead:

```
"/AT=Ralph/TN=(714) 624-7907/"@printer.example.com
"/TN=(714) 621-8465/AT=John Doe/O=Example/"@printer.example.com
"/AT=Dan/O=Example/OU=MIS Dept./MS=XJ 614/"@printer.example.com
"/AT=Dan Doe/USERNAME=DAN/"@printer.example.com
```

26.6.3 Logging

If the `logging` keyword is specified for a printer channel, then the channel will log its activity to the logging files. However, the printer channel logs the first 32 characters of the print queue name in place of the enqueuing channel name in the log file. On OpenVMS, in place of the message size in the log file, the printer channel logs the print job's queue entry number thereby allowing system accounting records to be cross-referenced with the logging file; on UNIX, the message size field is always shown as 0.

26.6.4 A PostScript Printer Channel

While the printer channel is primarily intended for use with dumb printers such as line printers, it is flexible enough to support PostScript printers requiring raw PostScript (*i.e.,* PostScript printers which are not driven by a symbiont capable of formatting raw ASCII text for printing).

The sample preamble file `printer_setup.ps_sample`, provided in the PMDF table directory (*i.e.*, as `PMDF_TABLE:printer_setup.ps_sample` on OpenVMS or as `/pmdf/table/printer_setup.ps_sample` on UNIX), documents how to do this. As described in this file, simply create the option file *x*`_option` in the PMDF table directory, where *x* is the name of your printer channel (*e.g.,* "`printer`"), and in this file place four lines as shown below. On OpenVMS:

```
PREAMBLE=@PMDF_TABLE:printer_setup.ps_sample
START_ATTRIBUTE=DO_HEADING
START_HEADERLINE=DO_HEADER
END_COVER=EJECT DO_BODY
```

On UNIX:

```
PREAMBLE=@/pmdf/table/printer_setup.ps_sample
START_ATTRIBUTE=DO_HEADING
START_HEADERLINE=DO_HEADER
END_COVER=EJECT DO_BODY
```

Owing to the design of the sample file `printer_setup.ps_sample`, it is necessary that there be a space at the end of the lines beginning with `START_ATTRIBUTE` and `START_HEADERLINE`.

26.6.5 Security Considerations

The printer channel is a secure program. However, there are some facets of its operation of which system managers should be aware.

Of paramount importance is to never configure a printer channel to print to an execution queue of some sort. Doing otherwise could cause a serious security hole under the right (wrong) circumstances. Consider the case where the channel doesn't require special printer control sequences and is configured to discard all message headers. Then, all that would be printed would be the body of a message directed to it. In that case, a message containing just commands appropriate to the execution queue could then be sent by anyone with permission to send mail to the printer channel.

Additional, operating system specific issues are discussed in the following sections.

26.6.5.1 Security Considerations on OpenVMS Systems

On OpenVMS systems, the ability to use the USERNAME addressing attribute is disabled by default. This prevents a user from maliciously generating print requests under any username. However, the USERNAME addressing attribute is useful since it allows specification of the OpenVMS username under which to print messages to a given address. That in turn both helps in accounting for print jobs and in identifying for which user the printer output is intended; (*e.g.,* given a username, the printer symbiont can display that username in trailer and flag pages). Use the SET_USERNAME option to enable the use of the USERNAME addressing attribute if you consider it safe for use at your site. Note that the account under which the printer channel runs will require CMKRNL privilege in order to submit print jobs under a username different than its own. The channel itself does not use that privilege: it is needed because the $SNDJBC system service checks for it.

26.6.5.2 Security Considerations on UNIX Systems

On UNIX systems, each message is printed by forking a child to execute a print command. The `system()` routine is used to accomplish this task; the child runs as the `pmdf` account. The format of the print command is specified with the PRINT_COMMAND channel option. Absolutely no user supplied data is put into the print command and hence there is no possibility for the channel to execute a user generated command. The only information substituted into the print command is the printer name and the name of the file to print. The printer name was specified when the channel was configured; the file name is that of a PMDF message file and produced by the channel itself. Moreover, that information which is substituted into the command is filtered so that any shell metacharacters are literalized. That is, all occurrences of the characters

```
" # $ & ' ( ) * ; < = > ? [ \ ] ` { | })
```

are preceded with a backslash, `\`. The QUOTE_CHARS channel option can be used to specify which characters, if any, require quoting.

26.7 Processing and Reprocessing Channels

The processing and reprocessing channels are essentially the intersection of all other channel programs — they perform only those operations that are shared among all other channels. In other words, such a channel is simply a channel queue whose contents are processed and requeued to other channels. Messages receive no special processing whatsoever.

The different between a reprocessing channel and a processing channel is that a reprocessing channel is normally “invisible” as a source or destination channel, as for instance in a CONVERSION, CHARSET-CONVERSION, or SEND_ACCESS mapping table, or in a source channel or destination channel specific rewrite rule. A processing channel, on the other hand, is visible like other PMDF channels.

It can appear that such a channel is effectively useless, but this turns out to be untrue. For example, the act of expanding a large mailing list can be very time-consuming. Timeouts can occur if this is done during the operation of a channel slave program with an open network connection. So PMDF provides the `expandlimit` channel keyword, which forces requeuing of the message to the reprocessing channel. Address expansion is then done as the reprocessing channel runs, free of any network timing constraints.

If a message destined to an address of the form *user@domain* is routed to the reprocessing channel, (*e.g.*, due to rewrite rules or the `expandlimit` keyword) then the reprocessing channel will simply re-enqueue the message to the channel associated with the domain *domain*; if a message destined to an address of the form *user@reprocessing-domain* is routed to the reprocessing channel, (*e.g.*, as can be the case for mailing lists using deferred expansion), then the reprocessing channel will re-enqueue the message to the local channel. In either case, the reprocessing channel performs any necessary expansion of the *user* part of the address.

When a PMDF channel has to generate a notification (bounce) message, such a notification message is initially enqueued to the processing channel.

A processing channel and a reprocessing channel are produced automatically by the PMDF configuration generator.[b]

26.7.1 Process Channel Definition and Rewrite Rules

If your configuration was generated by the PMDF configuration utility, then you do not need to add a reprocessing channel to your configuration: this was done automatically for you by that utility.[c] See the appropriate edition of the *PMDF Installation Guide* for instructions on using the configuration utility.

The first step in installing a processing channel is to insert the channel entry in PMDF's configuration file. The entry should have the form:

```
process
PROCESS-DAEMON
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the processing channel. Something like

```
process                     $U%process.localhostname
process.localhostname       $U%process.localhostname@PROCESS-DAEMON
```

where *localhostname* is the name of the local PMDF system will provide the necessary functionality. Once this is done any address of the form

[b] If you are using an older configuration generated prior to PMDF V5.2, you can need to manually add a processing channel definition and rewrite rules to your configuration. If you are using an older configuration generated prior to PMDF V5.0, you can need to manually add a reprocessing channel definition and rewrite rules to your configuration. For its own uses, PMDF will act as if process and reprocess channels are defined even if they are not explicitly present in your configuration. But if you want to make any site specific uses of such channels, explicitly addressing or rewriting to such channels, then you will need to have the channels explicitly present in your configuration.

[c] Configurations generated prior to PMDF V5.2 did not automatically include a processing channel.

user%host@process.*localhostname*

will be routed through the processing channel.

26.7.2 Reprocess Channel Definition and Rewrite Rules

If your configuration was generated by the PMDF configuration utility, then you do not need to add a reprocessing channel to your configuration: this was done automatically for you by that utility.[d] See the appropriate edition of the *PMDF Installation Guide* for instructions on using the configuration utility.

The first step in installing a reprocessing channel is to insert the channel entry in PMDF's configuration file. The entry should have the form:

```
reprocess
REPROCESS-DAEMON
```

Rewrite rules can be added if desired to make it possible to queue mail explicitly to the reprocessing channel. Something like

```
reprocess                       $U%reprocess.localhostname
reprocess.localhostname         $U%reprocess.localhostname@REPROCESS-DAEMON
```

where *localhostname* is the name of the local PMDF system will provide the necessary functionality. Once this is done any address of the form

user%host@reprocess.*localhostname*

will be routed through the reprocessing channel.

26.8 Generic SMTP Channels

The channel programs `test_smtp_master` and `test_smtp_slave` are provided as models upon which additional channels using the SMTP protocol can be built. They are intended as examples only and not as production channel programs.

Both programs require that the logical name PMDF_CHANNEL (on OpenVMS) or the environment variable PMDF_CHANNEL (on UNIX and NT) translate to the name of the channel they are servicing.

When `test_smtp_master` is executed, it looks in the queue cache database for messages waiting to be processed by the channel PMDF_CHANNEL. On OpenVMS, SMTP commands are written to SYS$OUTPUT and responses are expected on SYS$INPUT; on UNIX and NT, SMTP commands are written to `stdout` and responses are expected on `stdin`.

[d] Configurations generated prior to PMDF V5.0 did not automatically include a reprocessing channel.

Similarly, on OpenVMS, test_smtp_slave accepts SMTP commands on SYS$INPUT and writes responses to SYS$OUTPUT; on UNIX and NT, test_smtp_slave accepts SMTP commands on stdin and writes responses to stdout.

The distributed master.com command procedure on OpenVMS and the pmdf run utility and Job Controller on UNIX never invoke test_smtp_master and will have to be modified in order to use test_smtp_master. The code supporting tcp_master can be used as a model to drive test_smtp_master.

test_smtp_master includes code to distinguish between use as a direct connection to the target system and use for routing through a gateway. This facility parallels the gateway support found in TCP/IP channels, namely support for the daemon keyword.

26.9 DEC NOTES Channels (OpenVMS)

Note: The DEC NOTES channel is only supported on OpenVMS VAX and Alpha systems.

NOTES channels are used to provide a connection between PMDF and the DEC NOTES utility. DEC NOTES is a proprietary product of Hewlett-Packard Company; it is not a part of PMDF. This connection currently only works in one direction: mail messages will be registered as notes in DEC NOTES but DEC NOTES messages are not exported back out to PMDF.

NOTES channels try to convert mail messages into a format that is as close to a true NOTES posting as possible. In particular, options are provided to remove header information from messages or place it at the end of the posting. The NOTES channel program also attempts to group messages about a common subject into what DEC NOTES expects to see: a single original note with followup notes attached.

26.9.1 Setting Up the NOTES Channel

The first step in installing a NOTES channel is to insert the channel entry in PMDF's configuration file. The entry should appear as:

```
notes_local single
decnotes.domain.name
```

decnotes.domain.name should be a valid domain name that is reserved for use by the NOTES channel. One possible choice is to prepend the official local host name with "decnotes.". For example, in domain example.com, a reasonable domain name for the NOTES channel might be decnotes.example.com and the channel entry would then appear as

```
notes_local
decnotes.example.com
```

Some additional rewrite rules are also needed. Continuing the decnotes.example.com example, the appropriate rules would be:

```
decnotes                    $U@decnotes.example.com
decnotes.example.com        $U@decnotes.example.com
```

At this point the installation of the NOTES channel is basically complete. Any message sent to CONFERENCE@decnotes.example.com would be inserted into the conference named CONFERENCE. You can either subscribe addresses of this form to the mailing lists you want to read as notes files or you can set up aliases in the PMDF alias file and subscribe the aliases instead:

```
info-vax:           info-vax@decnotes.example.com
info-pmdf:          info-pmdf@decnotes.example.com
```

26.9.2 Using an Option File for the NOTES Channel

The NOTES channel processes e-mail messages converting them to DEC NOTES notes which are stored in a NOTES file. An options file can be used to control some of the characteristics of this conversion process. If an option file is used it must have the name `notes_local_option` and it must be located in the PMDF table directory, *i.e.*, `PMDF_TABLE:notes_local_option.` on OpenVMS. NOTES channel option files are in the same format used by other PMDF option files. See, for example, Section 26.4.1.5. The available options are:

NOTEFILE_*conference* (string, < 252 characters)
Normally the name of the NOTES file is derived from the name of the conference in an obvious way: the device specification `NOTES$LIBRARY:` is prepended and the file extension `.notes` is appended. In some cases NOTES files can be located elsewhere. The NOTEFILE_*conference* option provides a way to specify the name of a NOTES file for the conference *conference*. If it is specified it gives the full file specification; if it is not specified the defaults described above will be used instead.

PREFIXES (list of space-separated strings)
The PREFIXES option provides a default for cases where no specific PREFIXES_ *conference* option is provided. If not specified this option's value defaults to "Re: FWD:".

PREFIXES_*conference* (list of space-separated strings)
When subject grouping is enabled (see the discussion of SUBJECT_GROUPING_ *conference* options below) subject lines are compressed prior to being stored in subject files. Specifically, leading and trailing spaces are removed, multiple embedded spaces are converted into a single space, and leading prefixes often used in replies are removed. However, the specific leading prefixes used tends to be dependent on language and conference conventions.

The PREFIXES_*conference* options provide the means to specify what prefix strings should be removed on a per-conference basis. Prefix strings should be listed separated by single spaces. Case is ignored; entries can be specified in any case. If no PREFIXES_ *conference* option is specified the PREFIXES option is used as a default.

RETAIN_FAILURES (0 or 1)

The RETAIN_FAILURES option provides a default for cases where no specific RETAIN_FAILURES_*conference* option is provided. This option is provided mostly for debugging and is usualy set to the default value of 0.

RETAIN_FAILURES_*conference* (0 or 1)

When a delivery failure of some kind occurs on the NOTES channel the normal action is to abort the delivery and return the message to its sender. This default action can be disabled on a per-conference basis with the RETAIN_FAILURES_conference option. When this option is set to 1 all delivery failures will be retained in the channel queue. This is primarily used for debugging; it is useful to have messages retained in the queue until delivery quirks have been worked out.

The default value of the RETAIN_FAILURES_conference settings is obtained from the RETAIN_FAILURES option.

RETENTION_TIME (OpenVMS delta time)

The RETENTION_TIME option provides a default for cases where no specific RETENTION_TIME_*conference* option is provided. This option is normally set to whatever default a majority of conferences require. The time should be specified as an OpenVMS delta time string in the usual format (*i.e.,* DD HH:MM:SS; *e.g.,* RETENTION_TIME_GRIPES=30 00:00:00). The default value for this option is 30 days, RETENTION_TIME=30 00:00:00.

RETENTION_TIME_*conference* (OpenVMS delta time)

Entries in the subject grouping databases are only retained for a limited amount of time. This option provides a way of specifying how long to retain subject headers for the conference *conference*. The time should be specified as an OpenVMS delta time string in the usual format (*i.e.*, DD HH:MM:SS; *e.g.,* RETENTION_TIME_GRIPES=30 00:00:00). This option has no effect if subject grouping is disabled. If this option is not specified the default established by the RETENTION_TIME option will be used instead.

SET_PERSONAL_NAME (0-2)

The SET_PERSONAL_NAME option provides a default for cases where no specific SET_PERSONAL_NAME_*conference* option is provided. This option is normally set to whatever default a majority of conferences requires. The default value for this option is 1.

SET_PERSONAL_NAME_*conference* (0-2)

Each note that is posted contains a personal name field. This is normally the name of the person posting the note. When the NOTES channel posts a note it usually sets the personal name field to the address that's most appropriate for replies to go to. This address is given in a format that would be acceptable to VMS MAIL; that is, it has an IN% prefix and so forth.

In some cases, however, such a setting can be inappropriate. For example, some mailing lists can unconditionally appear to originate from a single user. In this case such a conversion is probably inappropriate.

The SET_PERSONAL_NAME_*conference* option controls this behavior for the conference *conference*. A value of 1 causes the conversion to take place. A value of 2 causes addresses specifically for replies to be ignored; the address of the actual message author will be used instead. A value of 0 blocks the conversion. If no option is specified the default is taken from the SET_PERSONAL_NAME option.

SUBJECT_GROUPING (-1 or 0 or 1)
The SUBJECT_GROUPING option provides a default for cases where no specific SUBJECT_GROUPING_*conference* option is provided. This option is normally set to whatever default a majority of conferences require. The default value for this option is 1.

SUBJECT_GROUPING_*conference* (-1 or 0 or 1)
The NOTES channel attempts to group messages with common subjects into a group of notes (specifically, an initial entry followed by one or more followup postings). This task is accomplished by building a database containing all the subject lines for recent messages and the corresponding note number. (A NOTES-SUBJECT mapping table can be used for even more control over the subject database entries; see Section 26.9.3.) A separate database is built for each conference; the names of the databases are controlled with the SUBJECTFILE_*conference* options. All this activity consumes resources and can in fact be inappropriate for some sorts of conferences. For example, the Risks-Digest conference is customarily grouped into messages containing multiple postings called digests; the names given to these digests are always unique and it would be a waste of time to try and group them in DEC NOTES.

The SUBJECT_GROUPING_*conference* option controls whether or not any attempt is made to group the messages sent to the conference *conference*. A value of 1 requests grouping. A value of 0 disables grouping. A value of -1 requests that the RETENTION_TIME_*conference* option (or failing that, the RETENTION_TIME option) be used to control grouping; messages received with the RETENTION_TIME setting will be collected in one group, but after RETENTION_TIME, a new group will be started. If this option is not specified the channel will obey the more general setting of the SUBJECT_GROUPING option.

SUBJECTFILE_*conference* (string, < 252 characters)
Normally the databases of subjects associated with each active conference *conference* have names of the form `PMDF_TABLE:` *`conference`*`.subjects` . This default can be overridden with an appropriate SUBJECTFILE_*conference* option. If this option is used, be sure to specify a complete path to the database.

USERNAME_*conference* (string, < 252 characters)
Each posting to a DEC NOTES conference is associated with a specific OpenVMS username. When the NOTES channel posts notes to a conference it normally uses the username it runs under; this is normally SYSTEM. In some cases, however, it can be useful to post notes under a different username. The USERNAME_*conference* option establishes a username to use for postings to the conference *conference*. This option must be set on a per-conference basis; the default if no option is given is to use the username the NOTES channel is running under (again, this is normally SYSTEM).

26.9.3 Further Control of the Subject Database

Normally, the NOTES channel builds a subject database with entries constructed from a message subject line and its corresponding note number. The option NOTES-SUBJECT mapping table can be used to modify what is used in such entries.

The format of the probe into the NOTES-SUBJECT mapping table, *i.e.*, the template, is

conference|*subject*

If the $Y flag is set in the mapping table pattern, then the pattern is used as the subject database entry.

For instance, the following NOTES-SUBJECT mapping table recursively strips off all but the last sequence number on a subject line for messages to the info-list conference:

```
NOTES-SUBJECT

  info-list|*$ [*]$ [*]              $R$Y$0$ [$2]
  info-list|*$ [*]                   $Y$0$ [$1]
```

27 The PMDF Queue to E-mail Symbiont (OpenVMS)

Note: The content of this chapter is only applicable to OpenVMS systems.

The queue to e-mail symbiont is a single threaded server symbiont which accepts, via the OpenVMS print subsystem command, messages to be sent as e-mail. Sending e-mail in this fashion has many inherent limitations and for this reason this symbiont is considered to be experimental at present; (*e.g.,* if the message is entered into the print queue via a remote system, it can not be possible to verify the name of the sender or to return error notifications).

Messages sent with the queue to e-mail symbiont are entered into the PMDF mail system through PMDF's callable SEND interface. Consequently, the behavior of the queue to e-mail symbiont is identical to that of the PMDF SEND utility described in the *PMDF User's Guide, OpenVMS Edition*.

The queue to e-mail symbiont can also be configured to use an addressing channel. Mail which lacks any forward addressing information (*e.g.,* `To:`, `cc:`, or `bcc:` addresses), can be passed on to an addressing channel. The addressing channel will then attempt to extract addressing information from the body of the message. Addressing channels are described in Chapter 26.

As described in Section 27.4, the symbiont can relay PostScript files to PMDF-FAX for transmission as facsimiles. This allows users of word processors to send FAXes directly from their word processing applications. This functionality works with Pathworks thereby enabling users of Macintoshes and PCs to print their documents from applications on their micro computers.

This symbiont is not part of the PMDF Process Symbiont and thus the installation of this symbiont does not require or depend upon the PMDF Process Symbiont.

27.1 Symbiont Configuration

Before creating a queue to e-mail queue, the symbiont executable must be copied to the system's executable directory, pointed at by the logical SYS$SYSTEM. (OpenVMS requires that all symbionts reside in the that directory.) To place a copy of the symbiont into this directory, issue the command

```
$ COPY PMDF_EXE:pmdf_q2email.exe SYS$SYSTEM:pmdf_q2email.exe
```

This will place the queue to e-mail symbiont in the system specific SYS$SYSTEM directory; to place it in the cluster common directory, use `SYS$COMMON:[sysexe]` in place of `SYS$SYSTEM:` in the above command.

A queue to e-mail queue is then created by initializing a server queue with pmdf_q2email as its processor on the appropriate cluster node, *node*:

```
$ INITIALIZE/QUEUE/DEVICE=SERVER/NOENABLE_GENERIC -
$_           /PROCESSOR=pmdf_q2email/ON=node:: queue-name
```

where *queue-name* is the name to use for the queue. If *node* is a member of a cluster, then omit the /ON= qualifier. The /NOENABLE_GENERIC qualifier prevents generic printer queues without specifically defined execution queues from unintentionally printing to this queue.

If the logical name PMDF_Q2EMAIL_LOG is defined and translates to a valid file name, then symbiont activity will be logged to that file. A new version of the file is created each time the symbiont is restarted.

27.2 Option Files

When the queue to e-mail symbiont queue is started, it consults the option file q2email_option. in the PMDF table directory, *i.e.*, PMDF_TABLE:q2email_option. If the file does not exist, the symbiont will use internal defaults. The format of the entries in the file is the same as those in other PMDF option files (but not the PMDF Process Symbiont). See, for instance, Section 7.2.

The available options are:

ADDRESSING_CHANNEL (text string <= 252 characters long)
The domain name (host name) associated with an addressing channel. Messages queued to the symbiont without any TO, CC, or BCC parameters will be forwarded to the specified addressing channel which will then try to extract information from the body of the message. There is no default for this option. This option applies to all instances of the queue to e-mail server and, for a given queue *queue*, can be overridden with the ADDRESSING_CHANNEL_*queue* option.

ADDRESSING_DELIMITER (single character)
Specifies the character to use as a command delimiter when generating a message containing addressing channel commands. Such messages are generated when a PostScript file is printed to the symbiont, in which case the symbiont attempts to extract addressing channel commands from the PostScript file itself and to reformat them for use by an addressing channel.

By default, a colon is used. When the target addressing channel uses a delimiter other than a colon (as specified with the DELIMITER option to that addressing channel), then the queue to e-mail symbiont must be made aware of this via the ADDRESSING_DELIMITER option.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the ADDRESSING_DELIMITER_*queue* option.

CHARSET
This option is used to specify the character set used in PostScript files passed to the symbiont. If this option is not specified, the local channel's character set will be assumed.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the CHARSET_*queue* option.

EXCLUDE_PROXIES (text string <= 252 characters long)

This option specifies a list of usernames which, when encountered, will be discarded and the print job's job name used instead. This option is intended for use with Pathworks which typically submits print jobs under the username MSAP$ACCOUNT or PCFS$ACCOUNT and specifies for the job name the remote Pathworks user's login name (*i.e.,* the username under which the remote user logged into the server). By using the job name in such instances, a more apt return address is used for the `From:` address of the e-mail message to be generated.

If not specified, the list "/MSAP$ACCOUNT/PCFS$ACCOUNT/" is used. The list must begin and end with a slash, "/", and a slash must delimit each entry in the list. No spaces or other punctuation can appear.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the EXCLUDE_PROXIES_*queue* option.

EXCLUDE_USERNAMES (text string <= 252 characters long)

A list of usernames to not use in a From: address (*e.g.,* DECNET). The address "<>" will be used in place of any username found in this list. If not specified, the list "/SYSTEM/DECNET/" is used. The list must begin and end with a slash, "/", and a slash must delimit each entry in the list. No spaces or other punctuation can appear.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the EXCLUDE_USERNAMES_*queue* option.

FROM_ALLOWED (0 or 1)

This option specifies whether or not users can specify the message's `From:` address with the FROM parameter. By default, users are not allowed to specify the `From:` address (FROM_ALLOWED=0). To allow users to set the `From:` address, specify `FROM_ALLOWED=1` in the option file.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the FROM_ALLOWED_*queue* option.

HEADERS_ALLOWED (0 or 1)

This option specifies whether or not users can specify an initial set of header lines for an e-mail message submission. (Users do so with the HEADERS parameter when they print a file to the symbiont.) By default, users are allowed to specify an initial set of header line (HEADERS_ALLOWED=1). To disallow users to specify header lines, specify HEADERS_ALLOWED=0 in the option file.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the HEADERS_ALLOWED_*queue* option.

SPACE_STRINGS (0 or 1)

By default (SPACE_STRINGS=1), when composing addressing channel commands from text strings extracted from a PostScript file, a space will be placed between each extracted string. This behavior can be inhibited by specifying SPACE_STRINGS=0.

This option applies to all instances of the the queue to e-mail server and, for a given queue *queue*, can be overridden with the SPACE_STRINGS_*queue* option.

27.3 Sending Mail with the Symbiont

Mail is sent with the queue to e-mail symbiont by printing a file with the OpenVMS `PRINT` command. Addressing information can either be specified with the `/PARAMETER` qualifier or embedded in the message body in a format acceptable to an addressing channel.

The recognized parameters are shown in Table 27–1.

Table 27–1 Queue to e-mail Symbiont Parameters

Parameter	Usage
BCC	List of one or more addresses to which to send a blind carbon copy.
CC	List of one more addresses to which to send a carbon copy.
ENCODING	Encoding format to use; can be one of BASE64, HEXADECIMAL, QUOTED_ PRINTABLE, or UUENCODE.
FROM	Address to use as the message's `From:` address; requires that the option FROM_ ALLOWED be set to 1 in the option file.
HEADERS	The symbiont is to use the RFC 822 headers already present in the file; does not take a value.
MODE	File access mode to use; can be one of CRATTRIBUTE, LFATTRIBUTE, CRLFATTRIBUTE, BLOCK, RECORD, or TEXT.
SUBJECT	Subject: line to use for the message.

For example, the `PRINT` command

```
$ PRINT/QUEUE=Q2EMAIL/PARAMETER=(TO="mrochek@example.com",-
_$    CC="bob@example.com,sue@example.com",SUBJECT="Test message") -
_$    message.txt
```

would send, via the queue Q2EMAIL, the file `message.txt` to mrochek@example.com, bob@example.com, and sue@example.com. The `Subject:` line will read "Subject: Test message".

To print a Word Perfect PostScript file `doc.ps` to the PS-FAX channel, a command of the following form might be used:

```
$ PRINT/QUEUE=Q2EMAIL/PARAMETER=(MODE=BLOCK,ENCODING=BASE64,-
_$    TO="""/fn=621 5319/at=Mrochek/""@ps-fax") doc.ps
```

When mail is to be sent from a remote system (*e.g.,* via LPD), then all of the addressing information can be embedded in the message body in a format acceptable to the addressing channel to which the message will be routed (since it will lack TO, BCC, or CC parameters specifying where to route it).

27.4 Printing Documents from Word Processors

The queue to e-mail symbiont has logic to extract addressing information from word processing documents printed as PostScript. These documents can either be printed locally or from network sources such as Pathworks. The first page of a document should contain addressing channel commands followed by a hard page break. The addressing information will be extracted from the PostScript in an application independent fashion and used to send the document along as e-mail. This allows users to print a document from their Macintosh, PC, *etc.*, and have the document, for instance, sent as a FAX via PMDF-FAX. The page containing addressing information will not appear in the FAXed document.

The logic used by the symbiont is as follows:

IF a file printed to the symbiont lacks `To:` addressing information *THEN*

IF the file is a PostScript file (*i.e.*, begins with "%!") *THEN*

1. Addressing information is extracted from the first page of the document, and
2. the message is converted to a MIME multipart/mixed message and passed on to the addressing channel. The converted message has the structure shown below:

```
... RFC822 message headers ...
Content-type: MULTIPART/MIXED; BOUNDARY="boundary"

--boundary
Content-type: TEXT/PLAIN; CHARSET=charset

... extracted addressing information ...

--boundary
Content-type: APPLICATION/POSTSCRIPT

... special PostScript to suppress the display of the
    first document page ...

--boundary
Content-type: APPLICATION/RMS
Content-transfer-encoding: BASE64

... PostScript document; encoded to prevent any possible damage
    (e.g., line wrapping) while transferring through
    the mail system ...

--boundary--
```

In 1. above, if the PostScript file does not follow the PostScript Document Structuring Conventions (DSC) then a second pass is made during which all PostScript strings are extracted. (In the first pass, the DSC is used to attempt to identify the beginning and end of the first page of the document.)

The character set information will be as specified with the `CHARSET` option. If no such option was specified, then the character set used by the local channel will be assumed.

27.4.1 Adding PostScript Support

To add support for the handling of PostScript documents, an option file must be created, and an addressing channel configured, if one is not already available. Consult Section 26.1 for details on configuring an addressing channel; consult Section 27.2 for instructions on setting up an option file.

Two options must be specified in the option file. (Others can or can not be required.) These two options are `ADDRESSING_CHANNEL=`*domain_name* and `FROM_ALLOWED=1`. Here, *domain_name* should be the domain name associated with the addressing channel you have or will configure; *e.g.,*

```
ADDRESSING_CHANNEL=address.example.com
FROM_ALLOWED=1
```

You can also need to specify the CHARSET option. If the addressing channel will be accepting PostScript from a Macintosh, then specify CHARSET=MACINTOSHPS; if the channel will be accepting PostScript from a PC running windows, then specify CHARSET=WINDOWSPS. See Section 27.4.2 for futher details on the use of this option.

A sample option file is shown below:

```
ADDRESSING_CHANNEL=address.example.com
CHARSET=MACINTOSHPS
FROM_ALLOWED=1
```

If multiple queues are set up, then use entries of the form

```
ADDRESSING_CHANNEL=address.example.com
CHARSET_queue1=MACINTOSHPS
CHARSET_queue2=MACINTOSHPS
FROM_ALLOWED=1
```

where *queue1* and *queue2* are the names of the individual queues.

27.4.2 Character Set Handling

The text encoded in PostScript files will typically be encoded in a platform dependent, or possibly even application dependent, character set. PMDF must know what character set is used so that it can properly convert the text extracted from the PostScript file to the HP MultiNational Character Set (DEC MCS) prior to processing by the addressing channel. For instance, Macintosh systems will use a character set in which an apostrophe (the ASCII character 27 in hexadecimal) will appear as the character with ordinal value D5; (in DEC MCS, this appears as the character Õ). This causes the addressing channel command

```
:Recipient's name: Fresnel
```

to appear in the PostScript file on VMS as

```
:RecipientÕs name: Fresnel
```

The CHARSET option is used to inform PMDF as to which character set is used in the PostScript. This option can specify any character set defined in the file `charsets.txt` in the PMDF table directory. Additional character sets can be added to that file as needed. When character sets are added to `charsets.txt`, the file must be recompiled and reinstalled:

```
$ PMDF CHBUILD
$ INSTALL REPLACE PMDF_CHARSET_DATA
```

Now, not only must PMDF be informed as to what character set is used, but it must also be told to perform character set conversions when processing mail queued to the addressing channel. To the mapping file add the table[2]

```
CHARSET-CONVERSION

  IN-CHAN=l;OUT-CHAN=address*;CONVERT         Yes
  IN-CHAN=l;OUT-CHAN=address*;IN-CHARSET=*    OUT-CHARSET=DEC-MCS
```

This will cause all messages queued to the addressing channel from the l channel (*i.e.,* the local channel) to be converted to HP MCS. (The queue to e-mail symbiont submits messages under the guise of the local channel.) See Chapter 6 for documentation on the use of the CHARSET-CONVERSION mapping table; see Chapter 5 for documentation on the use of the mapping file itself.

If you have a compiled configuration, then it must be recompiled before these entries in the mapping file will take effect. Likewise, the configuration needs to be recompiled whenever changes are made to the mapping file.

Note that a single queue to e-mail symbiont can only handle one type of incoming character set. Additional symbionts must be set up to handle additional character sets. This is not a limitation of the symbiont, but rather an inability to determine the input source to a symbiont. Different input sources can only be distinguished by using different printer queues which, in turn, requires multiple symbionts, one per printer queue. So, a separate printer queue should be set up for each set of input sources using a given character set. For instance, a site with both Macintosh and PC users should set up two queue to e-mail printer queues: one for the Macintosh users and one for the PC users. The `q2email_option.` file might then appear as

```
ADDRESSING_CHANNEL=address.example.com
CHARSET_queue_mac=MACINTOSHPS
CHARSET_queue_pc=WINDOWSPS
FROM_ALLOWED=1
```

where *queue_mac* and *queue_pc* are, respectively, the names of the Macintosh and PC printer queues.

[2] If you already have a CHARSET-CONVERSION table in the mapping file, then simply add the necessary table entries.

28 E-mail Firewalls and Other E-mail Security Considerations

This chapter discusses how to configure PMDF to act as an e-mail firewall, and various e-mail security issues to consider when doing so. Judicious implementation of the broad spectrum of techniques described below provides an effective e-mail firewall. Many of the techniques and configuration strategies described below can also be of interest even in a regular, non-e-mail-firewall configuration.

28.1 What is an e-mail Firewall?

Here an *e-mail firewall* refers to an enhanced, firewall-oriented e-mail handling component on an Internet firewall system. A basic Internet firewall system generally controls what TCP/IP interactions are allowed between the external world, considered to be unsafe, and an internal, protected environment, considered to be safe. To be an e-mail firewall system, this system should also check and control the e-mail passing between the internal and external environments.

- An e-mail firewall can perform address transformations, converting external presentation addresses in messages incoming from the external world to actual internal addresses, and transforming internal addresses to external presentation addresses on messages outgoing to the external world. See Chapter 3 for a discussion of centralized naming in general, and Section 28.4.8.4 below for mention of special considerations on an e-mail firewall.
- An e-mail firewall can enforce restrictions on what messages are allowed in or out. See Section 28.4.5 below. In particular, an e-mail firewall can disallow certain sorts of message traffic, and can be configured to protect against denial of service attacks.
- An e-mail firewall can be set up to perform filtering on message content, *e.g.*, limiting message size, or checking incoming binary attachments for viruses. See Section 28.4.7 below.
- An e-mail firewall is careful in what information it emits in response to external systems' possible probe attempts. See Section 28.4.8 below.
- And an e-mail firewall provides facilities for message logging and message traffic statistics. See Section 28.4.3.

28.1.1 The e-mail Firewall Orientation

One of the most important parts of setting up an effective e-mail firewall is having a security orientation: this is sometimes described as taking the attitude that "anything not permitted is forbidden".

There are a number of tradeoffs when configuring message handling. In a firewall configuration, the emphasis tends to be on tracking and control of messages and information passing through, whereas a regular PMDF-MTA configuration tends to emphasize efficiency and effectiveness. That is, where a regular PMDF-MTA configuration is geared towards "getting the mail through" one way or another, *e.g.*, accepting various address formats and fixing them up if necessary, a PMDF firewall configuration will typically be more concerned with ensuring that only "appropriate" addresses work and rejecting other addresses. A PMDF firewall configuration will typically maintain detailed logging information even at the expense of some additional overhead. And in a PMDF firewall configuration, there will typically be some concerns about what internal addressing information is exposed externally, which can mean performing additional work on address transformations, or stripping potentially useful (but overly informative) information from messages.

28.2 Preliminary Tasks Before Setting Up an e-mail Firewall

Before setting up an e-mail firewall, you should have an Internet firewall in place to control what sorts of general TCP/IP access are permitted to your systems, and you should consider and establish general security and e-mail messaging policies appropriate for your site.

28.2.1 Have an Internet Firewall in Place

Before setting up an e-mail firewall, you have presumably already set up a general firewall or "Internet firewall" to control general sorts of TCP/IP connections, etc., to your systems. Note that in comparison with remote logins or FTP access to files on your systems, e-mail is generally much less of an overall security exposure; there is generally not much point to concerning oneself with e-mail security until more fundamental security issues have been addressed. The discussion on e-mail firewalls in this chapter is referring solely to the additional control of e-mail that you may want to impose when you already have an Internet firewall set up.

28.2.2 Security and e-mail Policies

When setting up your Internet firewall, you presumably considered and established general security policies for your site. You should do the same for e-mail. For instance, depending upon your site, you may want to have explicit policies regarding e-mail address spoofing, the sending of harassing e-mail, list subscriptions, the sending of virus-infected PC executable programs, the use of e-mail for personal business, *etc.*

What is appropriate policy for your site will depend upon your site's goals and needs and what can be reasonably expected from your users. *Your greatest aid in good e-mail security, as in other security, is users who are educated as to your policies and committed to implementing them.* With the tightest security procedures in the world, if

your users do not understand the reasons for your policies and practices or find them overly burdensome, sooner or later some users will disregard or circumvent them.

28.3 The PMDF Firewall Configuration Utility

The PMDF CONFIGURE FIREWALL utility (OpenVMS) or `pmdf configure firewall` utility (UNIX) creates a basic PMDF firewall configuration. In accordance with the answers you give it about your particular site and goals, it uses the techniques described below to create a PMDF firewall configuration for your site. See the appropriate edition of the *PMDF Installation Guide* for instructions on using the utility and an example configuration.

28.4 Firewall Configuration Features

This section describes specific features and techniques useful in a firewall configuration.

28.4.1 Separating Message Traffic

One of the fundamental issues for a firewall configuration tends to be separation between internal and external messages: separating message traffic allows for tracking and appropriately controlling the different sorts of messages. So the first recommendation for a firewall system is to set up separate channels to handle messages originating from external sites versus messages originating from internal systems.

For background on rewrite rules, see Section 2.2; for background on channels, see Section 2.3.

28.4.1.1 Separating SMTP Over TCP/IP Message Traffic

The most common case is where messages originating from external sites come in to the PMDF system as SMTP messages over a TCP/IP channel. To separate the externally originating SMTP messages from internally originating SMTP message, use a separate TCP/IP channel in addition to the default TCP/IP channel; *e.g.*, use a `tcp_internal` channel in addition to the default `tcp_local` channel. Put the `switchchannel` channel keyword on your current `tcp_local` channel, the `allowswitchchannel` channel keyword (the default) on the `tcp_internal` channel and any other channels you want to allow switching to, and put `noswitchchannel` on all other channels, *e.g.*, the local channel, and use rewrite rules to associate internal TCP/IP system names and IP addresses with the `tcp_internal` channel and all other domains, *e.g.*, Internet domains, with the `tcp_local` channel.

When deciding whether and to what channel to "switch" an incoming connection, PMDF uses the literal IP address of the incoming connection to perform a reverse-pointing envelope rewrite looking for an associated channel. If that rewrite fails, or if that rewrite matches the default incoming TCP/IP channel, then PMDF will try rewriting the host name as found by a DNS reverse lookup on the incoming IP address; (depending on the use of any `ident*` keywords, such DNS reverse lookups can be disabled).

So in order to allow internal systems to be recognized as such, you should use IP literal rewrite rules to associate internal IP literals (at least during backwards envelope rewriting) with your internal TCP/IP channel. (Note that even if you do not normally use any IP literal rewrite rules so that PMDF tries to fall through to using host names, you should have such rewrite rules for your internal systems in case of DNS problems causing DNS reverse lookup failures.) If you want to limit the rewriting of internal IP numbers to actual system names in the forward direction, say if you do not want to allow external users to "probe" for internal IP number/internal system name correspondences, then you can want these IP literal rewrite rules to be backwards envelope specific, *i.e.*, `$E$R` rewrite rules.

Note that the default incoming TCP/IP channel is tcp_local and only system names or IP numbers recognized as internal system names are "switched" to the `tcp_internal` channel. This provides "failsafe" behavior; systems not specifically recognized (even internal systems, if the PMDF configuration has not been set up to recognize them) are handled by the external, "unsafe" channel.

By default, PMDF allows any channel to be "switched to"; *i.e*, the default is `allowswitchchannel`. On a firewall system in particular, it is likely appropriate to make `noswitchchannel` the default — for instance, you probably do not want to allow "switching" to the local channel—and mark only the specific channels for which you want to allow switching with the `allowswitchchannel` channel keyword.

28.4.1.1.1 Sample Configuration with Separate TCP/IP Channels

For instance, a site whose internal systems' IP numbers are all in the `[a.b.subnet]` range, might want channels

```
defaults noswitchchannel routelocal

l defragment ...
official-local-host-name

...

tcp_local single_sys smtp mx remotehost switchchannel inner
TCP-DAEMON

tcp_internal single_sys smtp mx noremotehost allowswitchchannel routelocal
TCP-INTERNAL
```

and rewrite rules

```
! Rewrite rules for private TCP/IP systems/domains
!
internaldomain1       $U%internaldomain1@TCP-INTERNAL
internaldomain2       $U%internaldomain2@TCP-INTERNAL
...
! Rewrite rules for private TCP/IP system/domain literals
!
[a.b.]                $U%[a.b.$L]@TCP-INTERNAL$E$R
...
! Rewrite rules for the Internet
!    Principality of Andorra
.AD                                       $U%$H$D@TCP-DAEMON
...
!    Zimbabwe
.ZW                                       $U%$H$D@TCP-DAEMON
```

Note also how the `remotehost` and `noremotehost` channel keywords are used on these channels. The `remotehost` and `noremotehost` channel keywords affect PMDF's handling of bare usernames ("addresses" that are illegally formatted in that they have no domain name). PMDF always inserts a domain name on such addresses, to make the addresses syntactically legal. For envelope `To:` addresses that are missing a domain name, PMDF always inserts the local host name. However for other sorts of addresses, such as `From:` addresses, another factor comes into play. The `remotehost` channel keyword on the `tcp_local` channel (handling incoming messages from external sites) tells PMDF to use the remote sending system's name (as determined by a reverse DNS lookup); the default `noremotehost` channel keyword on the `tcp_internal` channel (handling incoming messages from internal sites) tells PMDF to use its own local host name, which can be particularly appropriate in the case of poorly configured POP clients.

The `routelocal` keyword causes PMDF to attempt "short circuited" rewriting of any explicit routing in the address, such as `!` routing, `%-hack` routing, or `@` source routing. If a site expects no legitimate uses of explicit source routing, then blocking such usage blocks a potential way for external senders to relay messages by explicitly routing them past the firewall system through internal systems. Of course, sites that have legitimate uses of explicit routing will not be able to afford to block such usage and hence should not use `routelocal` on all channels.

The `inner` keyword causes PMDF address rewriting to be applied to addresses in embedded message parts (MESSAGE/RFC822 parts) within the message; if you are applying address reversal on outgoing messages, this is liable to be desirable.

28.4.1.2 The Case of an Internal Mailhub

In the case where the e-mail firewall relays all messages for internal systems to an internal mailhub system, and receives internal messages only from the internal mailhub, message traffic separation is straightforward: the connection to and from the mailhub system is the only internal connection, and all other connections are external.

Indeed, this can be thought of as a two system e-mail firewall setup, where the system we have been referring to as "the" e-mail firewall is the external portion of the e-mail firewall, and the mailhub system is the internal portion of the e-mail firewall. (In such a setup, particularly if the internal mailhub is a capable system such as another

PMDF system, you can even want to have the e-mail firewall system be ignorant of internal addressing details which are handled by the internal mailhub system.)

For such a setup you will want an internal channel for communicating with the internal mailhub, and an external channel (generally a TCP/IP channel, but UUCP, Phonenet, *etc.* channels are also possible) or channels for communicating with the rest of the world, and corresponding rewrite rules.

28.4.1.2.1 Sample Configuration With an Internal Mailhub System

For instance, for a site using SMTP over TCP/IP to communicate between the e-mail firewall and the mailhub, and with an SMTP over TCP/IP connection to the Internet, a configuration where all internal mail passes through the internal mailhub is simply a special (and potentially simpler) case of having separate TCP/IP channels, as in Section 28.4.1 above, with the major difference being that the tcp_internal channel should be a `daemon router` channel connecting to the mailhub system, *e.g.*,

```
defaults noswitchchannel routelocal

l defragment ...
official-local-host-name

tcp_local single_sys smtp mx remotehost switchchannel inner
TCP-DAEMON

tcp_internal smtp mx remotehost allowswitchchannel daemon router
mailhubdomain
TCP-INTERNAL
```

and rewrite rules

```
! Rewrite rules for private TCP/IP systems/domains
!
internaldomain1       $U%internaldomain1@TCP-INTERNAL
internaldomain2       $U%internaldomain2@TCP-INTERNAL
...
! Rewrite rules for private TCP/IP system/domain literals
!
[mailhubIPaddress]    $U%[mailhubIPaddress]@GTCP-INTERNAL$E$R
...
! Rewrite rules for the Internet
!    Principality of Andorra
.AD                                     $U%$H$D@TCP-DAEMON
...
!    Zimbabwe
.ZW                                     $U%$H$D@TCP-DAEMON
```

Compare this with the sample configuration excerpt shown in Section 28.4.1.1.1.

In this case, since messages from the internal side are coming from a PMDF system, the `remotehost` keyword is likely appropriate on both channels.

28.4.2 Postmaster Messages

On an e-mail firewall, the issue of postmaster mail can need a bit of extra consideration.

Domain names visible on the Internet are required to be able to accept mail addressed to `postmaster@`*`domain`*. PMDF itself will send warning messages and, (by default), copies of users' bounce messages to the postmaster.

It is critical that the postmaster address be a valid address for receiving mail. The normal recommendation for postmaster mail, therefore, is that it be directed to a local account (in the PMDF sense, i.e., delivered via the local channel) on the PMDF system itself. However, for an e-mail firewall it is possible that you will not want to have to have someone log on regularly to the e-mail firewall to check postmaster mail. If you do want mail to a different system, be sure to ensure that the connection between the e-mail firewall system and that other system is a very reliable connection; and be prepared that if something happens to break that connection, you will want to immediately change the postmaster address on the e-mail firewall to some other functioning address or be prepared for the potential for serious e-mail problems. (Bouncing postmaster mail is not pretty.)

28.4.3 Logging and Tracking Messages and Connections

This section points out some message logging and tracking techniques.

Snapshots of message traffic, and information on TCP/IP connections handled by the PMDF Service Dispatcher, are also available.

28.4.3.1 Logging Messages Passing through PMDF

The `logging` channel keyword causes PMDF write a log file entry for each pass of a message through a PMDF channel; see Section 2.3.4.84 for details. The `LOG_CONNECTION` PMDF option can be used to cause PMDF to log TCP/IP connections, such as SMTP, POP, and IMAP connections; such connection entries can either be included in the regular PMDF message log file, or written to a separate file. See Section 7.3.6 for discussion of the `LOG_CONNECTION` and `SEPARATE_CONNECTION_LOG` options. Note that with logging turned on, the cumulative `mail.log` file in the PMDF log directory will continue to grow and grow; PMDF itself never does anything with this log file and it is up to you to periodically write it to backup and delete it, or truncate it, or whatever your site prefers; the same is true for the `connection.log` file if TCP/IP connections are logged separately. Section 31.1.1 has a further discussion of managing the PMDF log files.

In addition to the base set of data logged when the `logging` keyword is used, there are options to cause the log output to include additional details, as discussed in Section 28.4.3.1.1 below.

28.4.3.1.1 Extra Logging Detail

In addition to the base set of logging enabled via the `logging` channel keyword, PMDF has options that cause additional information to be included in the entries written to the `mail.log*` files. Note that logging such additional information tends to incur additional overhead.

In particular, setting `LOG_MESSAGE_ID=1`, `LOG_CONNECTION=7`, and `LOG_FILENAME=1` in your PMDF option file can be of interest on a PMDF e-mail firewall. Logging the message ID makes it easier to find entries in the log file corresponding to a particular message, or to correlate different entries in the log file corresponding to a single message. Logging the SMTP client connection information can be useful to show just what system really sent the message to your PMDF firewall. Logging the process id also logs the thread id in the case of multithreaded channels; while the process ids themselves normally will be rather monotonous on a PMDF firewall system, being that of a Dispatcher Worker Process (for SMTP messages received) or a PMDF Process Symbiont process (for SMTP messages sent), having the process id and thread id logged is quite useful for correlating message entries with connection entries. Logging the filename can be useful if you want to correlate log file entries with actual message files currently in the PMDF queue area.

Setting `LOG_HEADER=1` can be of interest if you want to save certain message headers to the `mail.log*` files.

Additionally, setting `LOG_USERNAME=1` on a PMDF firewall system ought generally to result in fairly monotonous extra information being logged, as the username would normally just be the username of the user who last started the PMDF Service Dispatcher. Enable this options if you want to confirm that the username of processes enqueuing messages are as expected.

See Section 7.3.6 for more details on such logging options.

28.4.3.2 Snapshots of Message Traffic through PMDF

PMDF maintains channel counters based on the Mail Monitoring MIB, RFC 1566. These counters can provide "snapshots" of the state of the PMDF queues and a feel for the volume of messages passing through PMDF. See Chapter 31 for details.

28.4.3.3 Monitoring TCP/IP Connections to the Dispatcher

The PMDF Service Dispatcher maintains statistics on connections it handles, *e.g.*, the number of recent SMTP connections and the hosts from which the connections were made. See Section 11.7 for details.

28.4.4 Controlling Address Rewriting and Controlling Message Pathways

General PMDF configurations are usually set up to allow very flexible address rewriting, fixing up as many sorts of addresses as possible no matter what source the address comes in from. In a firewall configuration, however, you can want to control which sorts of rewriting happen for which sorts of messages. Therefore source and destination specific, and direction specific rewrite rules can be of particular interest. (This is akin/related to the point below regarding centralized naming, that a technique such as a directory channel (with a directory database), which isolates the address transformation to a particular channel, allowing for greater control at the cost of some additional overhead, can be more appropriate than the (more efficient but more "inline") alias database, general database, or mapping table sorts of approaches.)

For instance, consider a common setup where externally originating messages to internal users are expected to be addressed using a centralized format without internal node names, and where a directory channel (with a directory database) is then used to transform the addresses to the true internal address format. In a firewall configuration you can want to ensure not only that the centralized addresses work, but that *only* the centralized addresses work. So for instance, you might have a rewrite rule for the centralized domain routing it to the directory channel, and then make the rewrite rules for the true internal domains (routing such addresses to channels for sending internal) be source channel specific rewrite rules that only apply for messages coming from the `directory` channel.

28.4.4.1 Sample Configuration Controlling Internal Domain Rewriting

For instance, consider a site that wants to accept messages addressed in the form *First.Last*`@example.com`, route such messages to the `directory` channel where a directory database will transform the address to internal addresses such as *FLast*`@hosta.example.` or *Last*`@hostb.example.com`, or "*First Last*"`@ccmail.example.com`, but, for whatever reason, does not want to accept messages that come in from the external world already addressed to any of the domains `hosta.example.com`, `hostb.example.com`, or `ccmail.example.com`.

Note that unless the site has `MX` records for `hosta.example.com`, `hostb.example.com`, or `ccmail.example.com` pointing to the e-mail firewall system, then messages addressed using such explicit internal domain names would not normally ever reach the e-mail firewall system in the first place — *unless* the sender used explicit routing in the address, *e.g.*,

Last`%hostb.example.com@`*emailfirewalldomain*

To achieve the goal of routing messages addressed to `example.com` to the directory channel for expansion to internal addresses, but rejecting messages that come in from the external world already addressed to such an internal address, appropriate rewrite rules and channels might be along the lines of:

```
example.com             $U%example.com@DIRECTORY-DAEMON
hosta.example.com       $U@bogus$?Explicit domain addressing not allowed$Ntcp_local
hosta.example.com       $U%hosta.example.com@GTCP-DAEMON$Mdirectory
hosta.example.com       $U%hosta.example.com@GTCP-DAEMON$Mdefragment
hosta.example.com       $U%hosta.example.com@GTCP-DAEMON$Mconversion
hostb.example.com       $U@bogus$?Explicit domain addressing not allowed$Ntcp_local
hostb.example.com       $U%hosta.example.com@GTCP-DAEMON$Mdirectory
hostb.example.com       $U%hosta.example.com@GTCP-DAEMON$Mdefragment
hostb.example.com       $U%hosta.example.com@GTCP-DAEMON$Mconversion
ccmail.example.com      $U@bogus$?Explicit domain addressing not allowed$Ntcp_local
ccmail.example.com      $U%hosta.example.com@GTCP-DAEMON$Mdirectory
ccmail.example.com      $U%hosta.example.com@GTCP-DAEMON$Mdefragment
ccmail.example.com      $U%hosta.example.com@GTCP-DAEMON$Mconversion
...
.AD                 $U%$H$D@TCP-DAEMON
...
.ZW                 $U%$H$D@TCP-DAEMON
...

tcp_local ...
TCP-DAEMON

tcp_internal ...
GTCP-DAEMON

directory ...
DIRECTORY-DAEMON
```

28.4.5 Controlling e-mail Access

You can control which users can send to which users, what channels can send to what channels, and use hooks in PMDF to allow for dynamic, load-based rejection decisions.

28.4.5.1 Staticly Controlling e-mail Access

The PORT_ACCESS mapping table can be used to control from what IP numbers PMDF servers will accept connection attempts; the PMDF multithreaded SMTP server, POP3 server, IMAP server, and HTTP server check this table when a connection attempt comes in. The SEND_ACCESS and similar mapping tables can be used to control, based on From: address, To: address, and source and destination channel, what messages PMDF allows to pass through. See Section 21.2.1 for details on the PORT_ACCESS mapping table, and see Section 16.1 for details on the SEND_ACCESS and similar mapping tables.

For instance, consider a PMDF firewall system with postmaster address postmaster@example.com and with channels and rewrite rules set up, as described in Section 28.4.1, to segregate internal SMTP traffic onto a different channel than the tcp_local channel handling external SMTP traffic. On such a system, minimal PORT_ACCESS and ORIG_SEND_ACCESS mapping tables might be along the lines of:

```
PORT_ACCESS
  TCP|*|25|*|*                              $Y
  TCP|*|*|*|*                               $N
ORIG_SEND_ACCESS
  tcp_local|*|tcp_local|*                   $NRouting$ not$ allowed
  tcp_local|*|*|postmaster@example.com      $Y
  tcp_local|*|l|*                           $N
```

The first PORT_ACCESS entry shown allows incoming SMTP connections (*i.e.*, TCP connections to port 25) to PMDF's multithreaded SMTP server from anywhere. The second PORT_ACCESS entry shown disallows all other connections, *e.g.*, connections to the PMDF HTTP server, POP server, or IMAP server.

The first ORIG_SEND_ACCESS entry shown disallows other sites from routing SMTP mail by way of your system. (Note that it is normal and useful to allow explicit routing through your system. However, if you are worried about external users attempting to forge e-mail so that it appears to originate from your site, you can want to disallow the ability for routing through your system from external sites.) For a more detailed discussion of SMTP relay blocking, and examples of more sophisticated entries for blocking various source-routed address variants, see Section 16.1.6. The second ORIG_SEND_ACCESS entry is very important, as it allows messages to be sent to your postmaster account. The third ORIG_SEND_ACCESS entry disallows messages to any other (than the postmaster) local mailboxes on the PMDF firewall system.

28.4.5.2 Sidelining Messages for Manual Inspection

In some cases, it can be useful to allow in but sideline certain suspicious incoming messages (such as suspected junk e-mail)—perhaps messages to suspiciously large numbers of recipients or messages from particular suspicious sources—rather than simply rejecting them on either a temporary or permanent basis. The PMDF postmaster can then manually inspect the messages and determine the proper course: releasing the messages for delivery processing, bouncing them back to the original sender, or simply deleting them, if appropriate.

Note that allowing the messages onto the PMDF e-mail firewall allows them to use disk space on the PMDF system, where they will sit until the PMDF postmaster manually intervenes. So sidelining techniques should not be used unless the PMDF postmaster will in fact be checking periodically for the presence of such sidelined messages and taking action on them.

The holdlimit keyword on, for instance, the incoming TCP/IP channel can be used to automatically sideline messages to greater than a specified number of envelope recipients. See Section 2.3.4.16 for details.

Rather than rejecting messages (either permanently or temporarily), SEND_ACCESS or related mappings can be used to sideline incoming messages via the $H flag.

28.4.5.3 Dynamically Controlling e-mail Access, and Defending Against Denial of Service Attacks

A denial of service attack is where an attacker tries (intentionally or inadvertently) to overwhelm your system by flooding you with e-mail.

In some cases, adding a simple static entry to unconditionally reject messages from the problem address or site is a sufficient defense, particularly if you know ahead of time (or can quickly detect) that the attack is occurring; see Section 28.4.5.1 above. In other cases, however, you can either want to automate dynamic detection of message volume upswings sufficient to be considered an attack. Or you can not want to reject *all* messages from the problem address or site and instead want merely to "turn down the volume", *i.e.*, slow down the flow to a level more easily managed by your users or system. For instance, you can be under a practical or legal mandate to accept certain messages, or good messages can be mixed in with the bad message flow; in such a case turning down the volume to a manageable level allows the good messages a chance to get into your system while preventing the bad messages from overloading your resources.

The `PORT_ACCESS` and `SEND_ACCESS` mapping tables described in Section 28.4.5 above—as well as related mapping tables discussed in Section 16.1—can be used in more sophisticated ways than simple unconditional entries to achieve such goals, and can indeed be hooked into dynamic, heuristic routines to decide "Yea" or "Nay" on accepting messages, should you choose to provide such routines.

First, on the most simple level, the `PORT_ACCESS`, `SEND_ACCESS` or related mapping tables can take a random argument, effectively having PMDF "flip a coin" each time it needs to decide whether to accept a connection or message, respectively, or in the case of `SEND_ACCESS` and related mapping tables whether to sideline a message; see Section 5.3.2.5 for details.

For more sophisticated needs, the `PORT_ACCESS`, `SEND_ACCESS`, or related mapping tables can call out to site-supplied shareable image routines; see Section 5.3.2.10 for details. Such routines can, if you want, use PMDF API calls to access PMDF counters information; this can allow for heuristic decisions based on recent message load, comparing PMDF counters levels at one sampled time with PMDF counters levels when checked a little later; *e.g.*, "lots of messages came in to the `tcp_local` channel in the last few minutes, so let us reject additional connection attempts for the moment" or whatever decision basis you decide to implement.

VMS

On OpenVMS, it is also possible to capture a copy of PMDF's `mail.log_current` output in a mailbox device; so site supplied routines can use this information also, which is more detailed than the PMDF counters information, in making accept or reject decisions.

The heuristics for making dynamic decisions about accepting or rejecting messages tend to be very site specific, and involve a variety of critical issues. Note also that sites can want to keep the details of their own heuristic algorithms secure. Process Software recommends that sites interested in implementing their own denial of service prevention techniques obtain specialized consulting assistance.

Particularly when implementing dynamic rejection mechanisms, the TCP/IP channel options ALLOW_TRANSACTIONS_PER_SESSION and ALLOW_RECIPIENTS_PER_TRANSACTION can be of interest. The ALLOW_TRANSACTIONS_PER_SESSION option can be used to limit the number of messages accepted during a particular connection. After refusing a number of connection attempts from a particular site, once you do let them connect, they are liable to have a backlog of messages for your site which they will try to deliver during that connection. If you are attempting to "slow down" how much mail you accept from that site, you likely will want to use this option to say, in effect, "enough for now" after some point in the connection. Similarly, the ALLOW_RECIPIENTS_PER_TRANSACTION option can be used to limit the number of recipients allowed for a particular message; this can be useful in protecting against a denial of service attack in the form of messages blanketing large numbers of your users.

28.4.6 Controlling External Stimulation of Message Delivery

The extended SMTP command ETRN (RFC 1985) allows an SMTP client to request that a remote SMTP server start up processing of the remote side's message queues destined for sending to the original SMTP client; that is, it allows an SMTP client and SMTP server to negotiate "switching roles", where the side originally the sender becomes the receiver, and the side originally the receiver becomes the sender. Or in other words, ETRN provides a way to implement "polling" of remote SMTP systems for messages incoming to one's own system. This can be useful for systems that only have transient connections between each other, for instance, over dial-up lines. When the connection is brought up and one side sends to the other, via the ETRN command the SMTP client can also tell the remote side that it should now try to deliver any messages that need to travel in the reverse direction.

The SMTP client specifies on the SMTP ETRN command line the name of the system to which to send messages (generally the SMTP client system's own name). If the remote SMTP server supports the ETRN command, it will trigger execution of a separate process to connect back to the named system and send any messages awaiting delivery for that named system.

See also Section 2.3.4.33 and Section 2.3.4.34 for a general discussion of the SMTP ETRN command and PMDF channel keywords affecting PMDF's sending and behavior upon receipt of ETRN commands.

The ETRN command can be quite useful on an e-mail firewall system, particularly if communication partners have only dial-up or other intermittently scheduled connectivity. But for general external SMTP connections, you can want to limit the number of ETRN commands to which PMDF will respond in a single session, so that a single remote site cannot attempt to "monopolize" the PMDF system's message delivery processing. For this, the ALLOW_ETRNS_PER_SESSION channel option can be used in the external TCP/IP channel's option file; see Section 21.1.2.2.

Also, in the interest of limiting the amount of information about the firewall's configuration visible externally, you can want to block PMDF's normal echo of the name of the PMDF channel an ETRN command domain matches on the tcp_local channel handling general external SMTP connections. For this, specify the silentetrn channel keyword on the tcp_local channel.

28.4.7 Controlling e-mail Content and Message Priority

This section discusses imposing limits on the size or sensitivity of messages allowed through, and the related issue of setting message priority based on size, and general checking or filtering of message content.

28.4.7.1 Imposing Message Size Limits

The PMDF options BLOCK_LIMIT and LINE_LIMIT can be used to impose global size limits on all PMDF channels. The channel keywords `blocklimit` and `linelimit` can be used to impose size limits on specific destination channels; the channel keyword `sourceblocklimit` can be used to impose size limits on specific source channels.

The PMDF option `CONTENT_RETURN_BLOCK_LIMIT` can be used to force the `NOTARY` non-return of content flag for messages over the specified size; if such a message is subsequently bounced by a system that supports `NOTARY`, then the original message contents will not be included in the bounce message. The PMDF option `BOUNCE_BLOCK_LIMIT` can be used to cause PMDF, when generating a bounce message itself, to return only message headers for messages over the specified size.

28.4.7.2 Message Priority and Size Limits

On OpenVMS, PMDF jobs pay attention to message priority, *i.e.*, to the presence of a `Priority:` header in the message. The priority of message that PMDF immediate jobs (those jobs created when a message is first submitted) will handle can be controlled with the `immnonurgent`, `immnormal`, and `immurgent` channel keywords. The priority of message that PMDF periodic jobs (those jobs run periodically by PMDF to retry delivery of previously undelivered messages) will handle can be controlled with the `minperiodicpriority` and `maxperiodicpriority` keywords. Or the `urgentqueue`, `normalqueue`, and `nonurgentqueue` keywords can be used to cause messages of different priorities to be processed in different queues.

Some sites can want to control the time of day, for instance, at which low priority messages are sent. And note that the `nonurgentblocklimit`, `normalblocklimit`, and `urgentblocklimit` keywords can be used to forcibly downgrade the priority of "large" messages.

28.4.7.3 Imposing Message Sensitivity Limits

The channel keywords `sensitivitynormal`, `sensitivitypersonal`, `sensitivityprivate`, and `sensitivitycompanyconfidential` can be used to impose an upper limit on the sensitivity of messages that can be enqueued to a channel. For instance, a site wanting not to emit messages of Company-confidential sensitivity might choose to set `sensitivityprivate` on their channel that sends out to the Internet, generally a `tcp_local` channel. See Section 2.3.4.88 for more details.

28.4.7.4 Filtering Based on Message Headers

PMDF's channel level mailbox filter facility can be used to check the headers of incoming messages and make decisions to reject messages based on, for instance, the `Subject:` header. See Section 16.2 for details.

28.4.7.5 Checking or Filtering Message Content

The best protection against problematic message content coming into your site is educated users who are committed to implementing your site security policies. The best protection against problematic message content leaving your site is educated users who are committed to conforming to your site security policies. If the users want to evade your policies, they can generally work around any imposed restrictions, for instance, by encrypting their messages.

If you do want to check the actual content of message parts, the PMDF `conversion` channel can be useful. You can use a `CONVERSION` mapping table to direct that certain message traffic, that is messages coming in certain channels and going out certain channels, pass through the PMDF `conversion` channel. The PMDF `conversion` channel can then run whatever content checking or filtering procedure or utility you want.

For instance, some sites like to have binary message attachments checked by virus sniffing software. A `CONVERSION` mapping table along the lines of

```
CONVERSION
  IN-CHAN=*;OUT-CHAN=tcp_internal;CONVERT      Yes
```

and PMDF conversions file entries along the lines of

```
out-chan=tcp_internal; in-type=application; in-subtype=*;
  parameter-copy-0=*;
  command="yourviruscheckcommand 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=tcp_internal; in-type=audio; in-subtype=*;
  parameter-copy-0=*;
  command="yourviruscheckcommand 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=tcp_internal; in-type=image; in-subtype=*;
  parameter-copy-0=*;
  command="yourviruscheckcommand 'INPUT_FILE' 'OUTPUT_FILE'"

out-chan=tcp_internal; in-type=video; in-subtype=*;
  parameter-copy-0=*;
  command="yourviruscheckcommand 'INPUT_FILE' 'OUTPUT_FILE'"
```

where *yourviruscheckcommand* is a site-supplied command to do virus checking, will run any MIME message parts of type `APPLICATION`, `AUDIO`, `IMAGE`, or `VIDEO` MIME through your procedure.

Note that when you are using the conversion channel to check message parts on the PMDF firewall system, you are likely to want the `defragment` channel keyword on outgoing channels, particularly channels that send to internal systems. The MIME format allows for messages to be split into multiple pieces, which are normally not reassembled until arrival at the final destination system. However, if you want the

intermediate PMDF firewall system to check the message content, you will want to reassemble the message parts on the PMDF firewall system, so that the message content (rather than message content fragments) can be checked. See Section 2.3.4.76 for details.

28.4.7.6 Verifying Message Integrity

The `conversion` channel or service conversions can be used to perform site supplied message authentication (integrity) check procedures. See Chapter 6 for an overview of service conversions and the `conversion` channel. See also Chapter 23, discussing using BSMTP channels to "tunnel" messages between cooperating PMDF systems.

28.4.8 Restricting or Controlling Information Emitted

This section describes various ways information that you can not want to emit can "leak out" and describes ways of blocking this.

28.4.8.1 Restricting Access to PMDF Information via the PMDF HTTP Server

PMDF includes an HTTP server. This HTTP server is used to serve out PMDF version information, PMDF documentation, statistics on general PMDF operation (numbers of message moving through PMDF, *etc.*), and statistics on the Dispatcher's operation (IP addresses of connections, *etc.*). The HTTP server also provides a CGI interface to configuring PMDF mailbox filters, and CGI interfaces to the PMDF popstore for management, user access to their own popstore messages, and for users to change their own popstore passwords.

You should consider which, if any, of this information you want to allow access to from outside your site and which, if any, of this information you want to access on the PMDF e-mail firewall from within your site.

If you want to take advantage of absolutely none of this information even from within your site, then on the principle of "everything not permitted is forbidden" you can choose to simply disable PMDF's HTTP server entirely. To do so, edit your Dispatcher configuration file and remove or comment out the entire HTTP service definition section, see Section 12.1.1, and then restart the Dispatcher.

The more common case, however, is that you will want to allow access to at least some of the facilities from within your site: for instance, you will probably want to be able to access the PMDF monitoring information and mailbox filter configration from internal systems or at least your own workstation. You can even want to allow external access to a few selected facilities, such as the web interface to LDAP or X.500 directory information (if you are running an LDAP or X.500 directory which you want to be visible externally) or perhaps user-level access to the PMDF popstore[1] (if you are using the PMDF popstore to provide e-mail accounts for external users). In this case, you should

[1] User accounts are not generally implemented on an e-mail firewall system, but PMDF popstore accounts are a possible exception. For instance, PMDF popstore accounts might be set up specifically for use by users who are travelling out of the office.

make sure that your HTTP_ACCESS mapping is set up to allow only the access you want to permit, and to block all other access.

For instance, at a site whose internal addresses comprise the [1.2.3.0] subnet and where the PMDF HTTP server has been configured to run on its normal default port of 7633, then an HTTP_ACCESS mapping to allow full access to the PMDF HTTP server facilities from internal systems, allow access only to the PMDF popstore from external systems, and block all other access by external systems would be:

```
HTTP_ACCESS
! Allow full access from systems in the [1.2.3.0] subnet.
!
  $(1.2.3.0/24)|*|*|7633|*|*       $Y
!
! Allow access to user interfaces
! from external systems.
!
  *|*|*|7633|*|/msps_user/*       $Y
  *|*|*|7633|*|/chng_pwd/*        $Y
!
! Disallow all other access
!
  *                                $N
```

28.4.8.2 SMTP Probe Commands

During an SMTP connection, a remote sending side (or a person manually telnetting to your SMTP port) can issue commands requesting information such as a check on the validity of addresses. This very useful information can, however, be subject to abuse, *e.g.*, by automated search engines checking for valid email addresses on your firewall system. Therefore some sites can have an interest in disabling these helpful features.

Setting DISABLE_EXPAND=1 in your Internet TCP/IP channel disables the SMTP EXPN command. The SMTP EXPN command is normally used to expand (get the membership of) mailing lists.

Setting HIDE_VERIFY=1 in your Internet TCP/IP channel causes PMDF to return a "generic" response to the SMTP VRFY command. The SMTP VRFY command is normally used to check whether an address is a legitimate address on the local system. (Note that as it is required that SMTP servers support the VRFY command, PMDF has to return some sort of response; with HIDE_VERIFY=1, this response is simply a "maybe" sort of response rather than an explicit yes or no.)

Setting DISABLE_ADDRESS=1 in your Internet TCP/IP channel causes PMDF to disable responses to the PMDF SMTP server's private XADR command, which normally returns information about the channel an address matches.

Setting DISABLE_CIRCUIT=1 in your Internet TCP/IP channel causes PMDF to disable responses to the PMDF SMTP server's private XCIR command, which normally returns information about the PMDF message circuit checking facility.

Setting DISABLE_STATUS=1 in your Internet TCP/IP channel causes PMDF to disable responses to the PMDF SMTP server's private XSTA command, which normally returns information about the numbers of messages in PMDF queues.

Setting DISABLE_GENERAL=1 in your Internet TCP/IP channel option file causes PMDF to disable responses to the PMDF SMTP server's private XGEN command, which normally returns status information about whether a PMDF compiled configuration and character set are in use.

A sample TCP/IP channel option file to disable probing via the SMTP server, for a site using a tcp_local channel, would be as shown in Example 28–1.

Example 28–1 A Sample tcp_local_option File Disabling SMTP Probes

```
DISABLE_EXPAND=1
HIDE_VERIFY=1
DISABLE_ADDRESS=1
DISABLE_CIRCUIT=1
DISABLE_STATUS=1
DISABLE_GENERAL=1
```

See Section 21.1.2.2 for more details on TCP/IP channel options.

28.4.8.3 Internal Names in Received: Headers

Received: headers are normally exceptionally useful headers for displaying the routing that a message really took. Their worth can be particularly apparent in cases of dealing with apparently forged email, or in cases where one is trying to track down what happened to a broken messages, or in cases where a message does not appear to be repliable and one is trying to figure out who might know how to respond to the message. Received: headers are also used by PMDF and other mailers to try to detect message loops.

Message-id: headers are normally useful for message tracking and correlation.

However, on the converse side, Received: headers on messages you send out give the message recipient information about the routing that a message really took through your internal systems and tend to include internal system names and possibly an envelope recipient address. And Message-id: headers tend to include internal system names. At some sites, this can be considered a security exposure.

If your site is concerned about this information being emitted, first see if you can configure your internal systems to control what information they put in these headers. For instance, the PMDF options RECEIVED_DOMAIN and ID_DOMAIN can be used on a PMDF system to specify the domain name to use when constructing Received: headers and Message-id: headers, respectively. Although these options are not usually particularly relevant on the PMDF firewall system itself — after all, the firewall system is by definition a system whose name is intended to be visible to the outside world — if you have PMDF on internal systems also, the options can be of interest

on those internal PMDF systems. See Section 7.2 for details on these options. In a similar spirit, the channel keyword noreceivedfor can be used on channels on a PMDF system to instruct PMDF not to include the envelope recipient address in the Received: header it constructs, if limiting the exposure of internal routing addresses is a concern for your site. And for those rare cases where the inclusion of original envelope From: information in Received: headers constructed is of concern, the channel keyword noreceivedfrom can be used on channels on a PMDF system to instruct PMDF not to include envelope From: information in Received: headers it constructs in those cases (involving changing the envelope From:, such as certain sorts of mailing list expansions) where PMDF would normally include the envelope From: address.

If necessary, address reversal on the PMDF firewall system can be used to "canonicalize" message id's, to remove undesired information, (though note that this removal of information can mean that the resulting message id's are no longer particularly useful). Note that the USE_REVERSE_DATABASE PMDF option (in the option.dat file) must have bit 6 (value 64) set in order for address reversal to apply to message id's; for instance, if the option was previously set to the default value of 5, it must be set to 69 to apply to message id's. For instance, a site example.com that wants to ensure that no host.example.com domains appear in message id's might use a REVERSE mapping such as:

```
REVERSE

  *@*.example.com       $C$:I$0@example.com$Y$E
```

This REVERSE mapping only applies to message id's, due to the $:I flag.

As to Received: headers, only if you cannot configure your internal systems to control such sorts of information should you consider resorting to stripping such headers off entirely. Received: headers should not be removed lightly, due to their many and important uses, but if the internal routing and system name information in them is sensitive for your site and if you cannot configure your internal sytems to control what information appears in these headers, then you can want to strip off those headers on messages going out to the Internet via header trimming on your outgoing TCP/IP channel.

Note: Do not remove Received: headers or remove or simplify Message-id: headers on general principles or because your users do not like them. Removing such headers, among other things, (1) removes one of the best tracking mechanisms you have, (2) removes information that can be critical in tracking down and solving problems, (3) removes one of the few (and best) warnings of forged mail you can have, and (4) blocks the mail system's ability to detect and short-circuit message loops. Only remove such headers if you *know* your site *needs* them removed.

To implement header trimming, put the headertrim keyword — you will probably want the innertrim keyword as well — on your outgoing external TCP/IP channel or channels, generally tcp_local and possibly other tcp_* channels (possibly every tcp_* channel except your internal channel, tcp_internal), and create a header trimming file for each such channel. The headertrim keyword causes header trimming to be applied to the outer message headers; the innertrim keyword causes the header trimming to be applied also to embedded message parts (message/rfc822 parts) within the message. A sample header trimming file for a site using a tcp_local channel is shown in Example 28–2.

Example 28–2 A Sample `tcp_local_headers.opt` File for Stripping Received: Headers

```
Received: MAXIMUM=-1
MR-Received: MAXIMUM=-1
X400-Received: MAXIMUM=-1
```

See Section 2.3.4.59 for more details on header trimming.

28.4.8.4 Centralized Naming and Internal Addresses

One function that is often performed on an email firewall is the transformation of addresses from true, internal format to an external "centralized naming" format, *e.g.*, from *mailbox*`@`*host*`.example.com` to *First.Last*`@example.com`. (Note that if you have a "smart" internal mailhub system, *e.g.*, another PMDF system, you can choose to perform the centralized naming there, rather than on the e-mail firewall.) PMDF has flexible and varied facilities for performing such address transformations; see Chapter 3 for details. There are several points that can be of special interest when performing centralized naming on an e-mail firewall.

1. Put the `inner` keyword on (at least) your channels outgoing to the external world so that address rewriting will be applied to address in embedded message parts (message/rfc822 parts).

2. For the forward direction of address transformation, the directory channel with a directory database lookup offers the potential for more control via channel specific rewrite rules or a SEND_ACCESS mapping table than similar transformations performed via the PMDF alias file, alias database, or general database. The separate channel processing step that the directory channel incurs allows both for more control, and incurs more overhead, than the "inline" transformations performed via the alias file, alias database, or general database.

   The directory channel with directory database lookup, being a separate channel processing step rather than being performed "inline", also provides more "insulation" of the forward direction of address transformation from any potential for external SMTP probing and allows for more precise control with channel specific rewrite rules, should you have a use for them; the flip side of this is that in some cases PMDF will have to accept a message onto the system and then do a separate channel run to discover that an address is bad, rather than being able to reject a message immediately during the actual SMTP dialogue before the message ever comes onto the PMDF system (using PMDF system resources).

3. If you want to do LDAP or X.500 or CCSO directory lookups from the e-mail firewall system, say as part of your centralized naming scheme, note that either of these involves a network access to the directory. You should either put a good deal of effort into securing that network connection, to prevent spoofing at that network level, or alternatively, instead do the LDAP or X.500 or CCSO directory lookup from an internal "secure" system.

4. If you do not want notification messages generated by the e-mail firewall system to include the internal address, then you can want to use the `suppressfinal` keyword; see Section 2.3.4.27.

28.5 Other Features and Techniques that can Impact an e-mail Firewall

This section describes additional specific features and techniques that can impact an e-mail firewall.

28.5.1 General Performance Issues on an e-mail Firewall

Chapter 32 has a general discussion of tuning PMDF performance. This section is to point out in particular two of the issues discussed therein.

Perhaps even more than on a general PMDF system, you can want to consider using the `queue` channel keyword to segregate different channels' message processing to different queues, to ensure that particularly heavy traffic over one channel will not impact message traffic over another channel; *e.g.*, you can want to have one queue dedicated to your tcp_local channel and a different queue dedicated to your tcp_internal channel.

If the e-mail firewall system is not a system logged into regularly and thus subject to longer than usual spells where no one is actively checking it, thus potentially large buildups of backed up messages, then use of the `subdirs` channel keyword to split message files among multiple subdirectories can be a particularly good idea, to postpone the time at which OpenVMS RMS or UNIX directory performance begins to suffer due to large numbers of files in a single directory.

28.5.2 Additional Channel Keywords

Firewall considerations for additional channel keywords:

- The `deferred` channel keyword is discussed in Section 2.3.4.19; note that use of this keyword allows users to use the PMDF system's e-mail disk space as an extension of their own disk space. The default behavior of `nodeferred` is likely particularly appropriate behavior for an e-mail firewall if you are concerned about preventing a denial of service attack on your disk space.
- The `maxprocchars` channel keyword causes PMDF to not bother processing addresses in headers over the specified number of characters; in such a case PMDF passes the unprocessed headers through unchanged. See Section 2.3.4.79 for details.
- On OpenVMS, the `network` rightslist identifier requires users of the channel to have the NETMBX privilege; see Section 2.3.4.89 for details.
- The `nonurgentblocklimit`, `normalblocklimit`, and `urgentblocklimit` channel keyword can be used to set thresholds for when to downgrade message priority. Message priority, in turn, can affect whether PMDF attempts to send a message immediately, or whether PMDF lets the message wait until the next time the PMDF periodic delivery job runs. Depending upon your needs and circumstances, having large messages wait on the firewall system until a periodic job runs can be either

beneficial, by not using resources that could be used to send small messages immediately, or can expose your firewall system to the potential for getting its disk space "clogged up" with pending large messages in between runs of the periodic job.

- The `nox_env_to` channel keyword, which is the default, ensures that no X-Envelope-to: headers are put on messages. If you are using centralized naming, this is likely a particularly appropriate default. The `x_env_to` channel keyword, on the other hand, requests that PMDF include an X-Envelope-to: header.

28.5.3 Rightslist Identifiers and Group ids

The SEND_ACCESS and related mapping tables provide a general and flexible way to control who can send to whom. However, the use of rightslist identifiers (OpenVMS) or group ids (UNIX) to control who can send to whom is another possible approach. See Section 2.3.4.89 for details.

For instance, give the account under which PMDF runs — normally the SYSTEM account on OpenVMS — a rightslist identifier no one else has and put that identifier on all channels. Then only SYSTEM can use PMDF.

Or as another example, with a directory channel setup as above in Section 28.4.4.1 where one goal is to reject mail originally addressed to internal addresses, put a rightslist identifier on tcp_internal and grant that to SYSTEM, but have a different account run the SMTP server. Then at the rightslist identifier level also, external users cannot send straight to internal addresses (straight to the tcp_internal channel).

28.5.4 PMDF Options

The following describes special considerations for the following options on an e-mail firewall. For a detailed general discussion of these options, see Chapter 7.

- ACCESS_ERRORS: Note that if you do use rightslist identifiers (OpenVMS) or group ids (UNIX) for PMDF channel usage control, then the default ACCESS_ERRORS=0 is almost certainly desirable.
- BLOCK_LIMIT, BLOCK_SIZE, LINE_LIMIT: See also the discussion in Section 28.4.7.1 above.
- LOG_CONNECTION, LOG_FILENAME, LOG_FORMAT, LOG_MESSAGE_ID, LOG_SNDOPR, LOG_USERNAME: See also the discussion in Section 28.4.3.1 above.
- MAX_HEADER_BLOCK_USE and MAX_HEADER_LINE_USE: These options are used to control when PMDF automatically fragments messages. If you are using a conversion channel for message filtering, message fragmentation and defragmentation is an issue to consider.
- MAX_LOCAL_RECEIVED_LINES, MAX_MR_RECEIVED_LINES, MAX_RECEIVED_LINES, MAX_TOTAL_RECEIVED_LINES, MAX_X400_RECEIVED_LINES: These options specify after how many Received: lines PMDF decides that a message is looping. Once PMDF has decided that a message is looping, the message is renamed

to a `.HELD` file and sidelined, thereby breaking the cycle of the looping, until the PMDF system manager intervenes manually. See Section 28.4.8.3 above for a discussion of Received: headers. Adjusting these values can affect which system (if any) detects that the message is looping and sidelines it, which can be of interest if the PMDF e-mail firewall is not a system checked regularly.

- The NON_URGENT_BLOCK_LIMIT, NORMAL_BLOCK_LIMIT, and URGENT_BLOCK_LIMIT PMDF options can be used to set global thresholds for when to downgrade message priority. Message priority, in turn, can affect whether PMDF attempts to send a message immediately, or whether PMDF lets the message wait until the next time the PMDF periodic delivery job runs. Depending upon your needs and circumstances, having large messages wait on the firewall system until a periodic job runs can be either beneficial, by not using resources that could be used to send small messages immediately, or can expose your firewall system to the potential for getting its disk space "clogged up" with pending large messages in between runs of the periodic job.
- NAME_TABLE_NAME: Using logical names for address transformations is not recommended in any case, as it allows senders to "probe" for logical names, and is distinctly not a good idea on an e-mail firewall. The default where this option is not specified is strongly recommended.
- RETURN_ADDRESS: If you are directing postmaster mail to somewhere other than the PMDF e-mail firewall system itself, then you can want to consider setting this option to match. However, consider with caution: as noted in the discussion above, using such a non-local address (and in particular, changing the return address used on messages *from* the postmaster by setting this option) can lead to rapid mail looping and pile-ups of huge numbers of spurious error messages.
- RETURN_DELIVERY_HISTORY and HISTORY_TO_RETURN: These control how much information PMDF includes in returned messages about why the message could not be delivered. This can be very useful information to the recipient of the returned message, but in some cases this helpful information canc include information that you prefer recipients not see, *e.g.*, internal system names.
- REVERSE_ENVELOPE: Note that if you are performing address reversal, say with a REVERSE mapping or reverse database, then the default REVERSE_ENVELOPE=1 is almost certainly desirable.

Volume III

The *PMDF System Manager's Guide* is in four volumes. Volume I comprises Chapter 1 through Chapter 13. Volume II comprises Chapter 14 through Chapter 28. Volume III comprises Chapter 29 through Chapter 34.

PMDF software products are marketed directly to end users in North America, and either directly or through distributors in other parts of the world depending upon the location of the end user. Contact Process Software for ordering information, to include referral to an authorized distributor where applicable:

Process Software, LLC
959 Concord Street
Framingham, MA 01701 USA
+1 508 879 6994
+1 508 879 0042 (FAX)
sales@process.com

29 Utilities on OpenVMS

PMDF contains a modest collection of management utility programs, which are used to perform various maintenance, testing, and management tasks. The following sections describe these utilities. Note that many of the utilities are mentioned elsewhere in this document in the context of how they are actually used. PMDF popstore and PMDF MessageStore utilities are described in the *PMDF popstore & MessageStore Manager's Guide*. User-level utilities are described in the *PMDF User's Guide*.

Briefly, the PMDF utilities, both those documented in the *PMDF User's Guide* or *PMDF popstore & MessageStore Manager's Guide* and those documented here, are shown in Table 29–1. Those utilities only available under OpenVMS are marked with a †; those only available under UNIX and NT are marked with a ‡; those available on UNIX but not available on NT are marked with a §.

Table 29–1 PMDF Utilities

Web-based utilities and displays	
URL	**Description**
`http://`*`pmdfhost`*`:7633/configure/`	Configure: generate PMDF configuration files; see the *PMDF Installation Guide*
`http://`*`pmdfhost`*`:7633/dispatcher/`	Dispatcher Statistics: view statistics on recent connections to the Dispatcher, *e.g.*, SMTP, POP and IMAP connections
`http://`*`pmdfhost`*`:7633/mailbox_filters/`	Mailbox Filters: generate and modify system and user mailbox filters controlling filtering of incoming messages
`http://`*`pmdfhost`*`:7633/qm/`	Message Queue Management: queue management utility
`http://`*`pmdfhost`*`:7633/msgstore/`	MessageStore Administration: manage PMDF MessageStore; see the *PMDF popstore & MessageStore Manager's Guide*
`http://`*`pmdfhost`*`:7633/monitor/`	Monitoring: view PMDF counters; on OpenVMS, also view the status of PMDF processing queues

Table 29–1 (Cont.) PMDF Utilities

Web-based utilities and displays	
URL	**Description**
`http://`*`pmdfhost`*`:7633/chng_pwd/`	Password Change Utility: change your e-mail password; usually used to change a PMDF MessageStore or PMDF popstore account password, but may also change a system password, depending upon configuration; see the *PMDF popstore & MessageStore Manager's Guide*
`http://`*`pmdfhost`*`:7633/popstore/`	popstore Administration: manage PMDF popstore; see the *PMDF popstore & MessageStore Manager's Guide*
`http://`*`pmdfhost`*`:7633/msps_user/`	popstore and MessageStore User Interface: change your password or view your account settings, or for popstore only view your messages; see the *PMDF popstore & MessageStore Manager's Guide*

	Command utilities			
	OpenVMS utility		**UNIX utility**	**Description**
†	`CACHE/CLOSE`			Have detached processes close their connections to the queue cache database
†	`CACHE/REBUILD`			Build a new, synchronized queue cache database
	`CACHE/SYNCHRONIZE`		`cache -synchronize`	Synchronize the current queue cache database
		‡	`cache -view`	View entries in the queue cache database
	`CHBUILD`		`chbuild`	Compile the PMDF character set conversion tables
	`CLBUILD`		`clbuild`	Compile a PMDF command definition file
	`CNBUILD`		`cnbuild`	Compile the PMDF configuration, alias, mapping, security, system wide filter, circuit check, and option files
	`CONFIGURE`	§	`configure`	Create a PMDF configuration file
†	`convert_cache.com`			Perform a CONVERT/RECLAIM on the queue cache

§Not available on NT

†Available on OpenVMS only

‡Available on UNIX only

Table 29–1 (Cont.) PMDF Utilities

	Command utilities		
	OpenVMS utility	**UNIX utility**	**Description**
		§ `convertdb`	Read entries from a V6.0-V6.4 PMDF `crdb` database and write out a corresponding V6.5 or later PMDF `crdb` database
	`COUNTERS/CLEAR`	`counters –clear`	Clear the in-memory cache of channel counters
†	`COUNTERS/CRDB`		Create a database of channel counters
	`COUNTERS/SHOW`	`counters –show`	Display the contents of the database of channel counters
†	`COUNTERS/SYNCHRONIZE`		Synchronize the in-memory cache of channel counters with the database
	`COUNTERS/TODAY`	`counters –today`	Display PMDF's count of the number of messages processed today
	`CRDB`	`crdb`	Create a PMDF database
	`DB`	`db`	Manage a personal alias database; see the *PMDF User's Guide*
†	`DCF`		Convert WPS and DX files to ASCII; provided with PMDF-MR
	`DECODE`	`decode`	Decode a file encoded using MIME encodings; see the *PMDF User's Guide*
\\ `dumpdb` \	Dump entries in a PMDF `crdb` database to a flat text file)		
		® `edit`	Edit PMDF configuration files
	`ENCODE`	`encode`	Encode a file using MIME encodings; see the *PMDF User's Guide*
		‡ `find`	Find the filename corresponding to the specified "version" of a PMDF file
†	`FOLDER`		Place a message file into a VMS MAIL folder; see the *PMDF User's Guide, OpenVMS Edition*
†	`FORWARD`		Set a forwarding address in the PMDF alias database; see the OpenVMS Edition of the *PMDF User's Guide*
†	`G3`		Analyze a PMDF-FAX G3 file; provided with PMDF-FAX

®Available on NT only
§Not available on NT
†Available on OpenVMS only
‡Available on UNIX only

Table 29–1 (Cont.) PMDF Utilities

	Command utilities			
	OpenVMS utility		**UNIX utility**	**Description**
†	INSTALL			Install or deinstall PMDF images and databases
	KILL	§	kill	Kill the specified PMDF component
	LICENSE		license –verify	On OpenVMS, activate or deactivate PMDF bundle licenses on a node; on Solaris, Linux, and Windows, verify the validity of a PMDF license file
†	MAIL			An extended version of VMS MAIL; see the OpenVMS Edition of the *PMDF User's Guide*
	migrate	§	migrate	Copy message folders from one IMAP host to another IMAP host; see the appropriate edition of the *PMDF User's Guide*
	MOVEIN	§	movein	Migrate a user's mailbox from one message store to another; see the *PMDF popstore & MessageStore Manager's Guide*
	MSGSTORE		msgstore	Interactive PMDF Message Store management utility; see the *PMDF popstore & MessageStore Manager's Guide*
	PASSWORD		password	Set remote authentication passwords
	POPSTORE		popstore	Interactive PMDF popstore management utility; see the *PMDF popstore & MessageStore Manager's Guide*
	PROCESS	§	process	List currently running PMDF jobs
		§	profile	Set local user's choice of delivery mechanism in the PMDF user profile database
†	PS			Convert text and Runoff .mem files to PostScript; provided with PMDF-FAX; see the OpenVMS Edition of the *PMDF User's Guide*
		‡	purge	Purge PMDF log files
	QCLEAN		qclean	Hold or delete message files matching specified criteria

§Not available on NT

†Available on OpenVMS only

‡Available on UNIX only

Table 29–1 (Cont.) PMDF Utilities

Command utilities		
OpenVMS utility	**UNIX utility**	**Description**
QM	qm	Manage PMDF message queues; see also the web-based QM utility, Section 31.2
QTOP	qtop	Display the most frequently occurring strings found in message files in the PMDF queue area
RESTART	restart	Restart detached PMDF processes
RETURN	return	Return (bounce) a mail message to its originator
master.com	run	Process messages in a specified channel
SEND	send	Send a mail message; see the *PMDF User's Guide*
SHUTDOWN	shutdown	Shut down detached PMDF processes
STARTUP	startup	Start detached PMDF processes
submit_master.com	submit	Process messages in a specified channel
submit_master.com	submit_master	Process messages in a specified channel—on UNIX, a synonym for submit
tls_certdump	tls_certdump	Dump the contents of a certificate file
tls_certreq	tls_certreq	Generate a public key pair and a certificate request
tls_ciphers	tls_ciphers	List available ciphers
	‡ view	Display the specified "version" of a PMDF file
VERSION	version	Print PMDF version number

‡Available on UNIX only

This chapter is broken into two main sections. The first section, Command Line Utilities on OpenVMS, describes the command line utilities available on OpenVMS (except those documented in the *PMDF User's Guide*); the second section, Interactive Utilities on OpenVMS, describes the interactive QM utility available on OpenVMS.

29.1 Command Line Utilities on OpenVMS

This section documents the PMDF command line utilities available on OpenVMS; these utilities are implemented as DCL verbs using the CDU file `PMDF_COM:pmdf.cld`.

CACHE/CLOSE—Close queue cache I/O channels

Force detached PMDF processes to close any open I/O channels to the queue cache database.

SYNTAX

PMDF CACHE/CLOSE

Command Qualifiers	Defaults
None.	*None.*

restrictions SYSLCK privilege is required to in order to use this utility.

PARAMETERS

None.

DESCRIPTION

The CACHE/CLOSE utility is used to force, cluster-wide, all detached PMDF processes to close any open I/O channels that they have to the queue cache database. This is generally done for two reasons: to close all channels to the file so that it can be modified, and to force detached processes to re-open the queue cache database file so as to begin using any new version of that database.

Sites using the TCP/IP channel will have detached PMDF processes which may need to close channels to the database.[1]

EXAMPLES

After a new queue cache database is built with CACHE/REBUILD, a CACHE/CLOSE command should be issued to force any detached processes to begin using the new database:

```
$ PMDF CACHE/REBUILD
$ PMDF CACHE/CLOSE
$ ! ...wait a minute or two...
$ PMDF CACHE/SYNCH
```

[1] Also, customer-supplied detached processes which use the PMDF API routine `PMDF_set_call_back` may be notified of the need to close the queue cache database with the CACHE/CLOSE command.

CACHE/REBUILD—Build a new queue cache

Build a new, synchronized queue cache database.

SYNTAX

PMDF CACHE/REBUILD

Command Qualifiers	Defaults
None.	*None.*

restrictions Requires sufficient privileges to create a file in the PMDF_TABLE: directory, and to scan the PMDF_QUEUE:[*] directories.

PARAMETERS

None.

DESCRIPTION

The CACHE/REBUILD utility creates a new, synchronized queue cache database. Although the new database will inherit the ownership and file protections of the previous database, it is a good idea to check afterwards that the new queue cache is owned by the same UIC as the `queue.dir` and `log.dir` files in the `PMDF_ROOT:[000000]` directory and that the file is protected against group and world access (S:RWED,O:RWED,G,W).

Rebuilding the queue cache database with this command should only be performed as a last resort, *e.g.*, if disk problems have corrupted your queue cache database, as it will cause loss of some information from the queue cache database. (The sort of information lost includes, but is not limited to, message creation dates, message deferral dates, message expiration dates, and the original message owner information used by the PMDF QM utility to allow users to bounce their own messages.)

The command PMDF CACHE/CLOSE should be issued immediately after building the new queue cache so as to ensure that any detached processes close any I/O channels to the old database and open new channels to the new database.

The queue cache database is the file pointed at by the PMDF_QUEUE_CACHE_DATABASE logical. Normally, this is the file `queue_cache.dat` in the `PMDF_TABLE:` directory. This file should be protected against world and group access and be owned by the same UIC as the directory files `queue.dir` and `log.dir` in the `PMDF_ROOT:` directory.

EXAMPLES

To build a new queue cache database issue the commands

```
$ PMDF CACHE/REBUILD
$ PMDF CACHE/CLOSE
$ ! wait a minute or two
$ PMDF CACHE/SYNCH
```

CACHE/SYNCHRONIZE—Synchronize the queue cache

Update the queue cache database so as to reflect all messages currently present in the message queues.

SYNTAX

PMDF CACHE/SYNCHRONIZE

Command Qualifiers	Defaults
None.	*None.*

restrictions Requires sufficient privileges to scan the PMDF_QUEUE:[*] subdirectories and add entries to the queue cache database.

PARAMETERS

None.

DESCRIPTION

The CACHE/SYNCHRONIZE utility updates the active queue cache database by updating it to reflect all non-held message files currently present in the PMDF_QUEUE:[*] subdirectories.

The PMDF CACHE/CLOSE command does not need to be issued in conjunction with the PMDF CACHE/SYNCHRONIZE command.

The queue cache database is the file pointed at by the PMDF_QUEUE_CACHE_DATABASE logical. Normally, this is the file queue_cache.dat in the PMDF_TABLE: directory. This file should be protected against world and group access and be owned by the same UIC as the directory files queue.dir and log.dir in the PMDF_ROOT: directory.

EXAMPLES

To synchronize the queue cache, for instance after renaming a message file, issue the command

```
$ PMDF CACHE/SYNCHRONIZE
```

CHBUILD—Character set table compiler

Compile the PMDF character set conversion tables in an OpenVMS shareable image.

SYNTAX

PMDF CHBUILD

Command Qualifiers	Defaults
/IMAGE_FILE=file-spec	*/IMAGE_FILE=PMDF_CHARSET_DATA*
/MAXIMUM	*/NOMAXIMUM*
/OPTION_FILE=file-spec	*/OPTION_FILE=PMDF_CHARSET_OPTION_FILE*
/STATISTICS	*/NOSTATISTICS*

PARAMETERS

None.

DESCRIPTION

The CHBUILD utility compiles the character set conversion tables into an OpenVMS shareable image. The resulting file can then be installed with the OpenVMS INSTALL utility.

PMDF ships with very complete character set tables so it is not normally necessary to run this utility.

COMMAND QUALIFIERS

/IMAGE_FILE=file-spec
/NOIMAGE_FILE
By default, CHBUILD creates as output the file PMDF_CHARSET_DATA. With the /IMAGE_FILE qualifier, an alternate file name may be specified.

When the /NOIMAGE_FILE qualifier is specified, CHBUILD does not produce an output file. This qualifier is used in conjunction with the /OPTION_FILE qualifier to produce as output an option file which specifies table sizes adequate to hold the tables required by the processed input files.

/MAXIMUM
/NOMAXIMUM (default)
The file `PMDF_TABLE:maximum_charset.dat` is read in addition to the file PMDF_CHARSET_OPTION_FILE when /MAXIMUM is specified. This file specifies near maximum table sizes but does not change any other option file parameter settings. Only use this qualifier if the current table sizes are inadequate. The /NOIMAGE and /OPTION_FILE qualifiers should always be used in conjunction with this qualifier—it makes no sense to output the enormous character set image that is produced by /MAXIMUM, but it does make sense to use /MAXIMUM to get past size restrictions in order to build a properly sized character set option file so

that a properly sized character set data can be built with a subsequent CHBUILD invocation.

/OPTION_FILE[=file-spec]
/NOOPTION_FILE (default)
CHBUILD can optionally produce an option file that contains correct table sizes to hold the conversion tables which were just compiled (plus a little room for growth). The /OPTION_FILE qualifier causes this file to be output. By default, this file is the file pointed to by the logical PMDF_CHARSET_OPTION_FILE, normally `PMDF_TABLE:option_charset.dat`. The value on the /OPTION_FILE qualifier may be used to specify an alternate file name. If the /NOOPTION_FILE qualifier is given, then no option file will be output.

CHBUILD always reads any option file that is already present via the PMDF_CHARSET_OPTION_FILE logical name; use of this qualifier will not alter this behavior. However, use of the /MAXIMUM qualifier causes CHBUILD to read options from `maximum_charset.dat` in addition to PMDF_CHARSET_OPTION_FILE. This file specifies near maximum table sizes. Only use this qualifier if the current table sizes are inadequate, and only use it to create a new option file. The /NOIMAGE qualifier should always be specified when /MAXIMUM is specified since a maximum-size image would be truly enormous and extremely wasteful.

/SIZES
/NOSIZES (default)
The /SIZES qualifier instructs PMDF CHBUILD to output information on the sizes of the uncompiled character set tables.

/STATISTICS
/NOSTATISTICS (default)
The /STATISTICS qualifier instructs CHBUILD to output information on the compiled conversion tables. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the /OPTION_FILE qualifier is needed.

EXAMPLES

The standard commands used to compile character set conversion tables and reinstall them are:

```
$ PMDF CHBUILD
$ INSTALL REPLACE PMDF_CHARSET_DATA
```

CLBUILD—Command definition compiler

Compile a PMDF command definition file into an OpenVMS shareable image.

SYNTAX

PMDF CLBUILD *cld-file-spec*

Command Qualifiers	Defaults
/DEBUG	*/NODEBUG*
/IMAGE_FILE=file-spec	*/NOIMAGE_FILE*
/MAXIMUM	*/NOMAXIMUM*
/OPTION_FILE=file-spec	*/NOOPTION_FILE*
/SIZES	*/NOSIZES*
/STATISTICS	*/NOSTATISTICS*

PARAMETERS

cld-file-spec
The file specification of a PMDF command line definition file to read as input, *e.g.*, `PMDF_COM:pmdf.cld`

DESCRIPTION

The `CLBUILD` utility compiles a command line definition file into an OpenVMS shareable image. The resulting image can then be installed with the OpenVMS `INSTALL` utility.

PMDF ships with a pre-compiled command line definition image so it is not normally necessary to run this utility.

COMMAND QUALIFIERS

/DEBUG
/NODEBUG (default)
The `/DEBUG` qualifier causes `CLBUILD` to output debug information regarding its operation.

/IMAGE_FILE=file-spec
/NOIMAGE_FILE (default)
By default, `CLBUILD` does not produce a compiled command definition file. In order to produce a compiled command definition file, the file to produce must be specified using the `/IMAGE_FILE` qualifier. Note that the logical name `PMDF_COMMAND_DATA` may be specified as the image file-spec, if the goal is to produce a compiled version of the main PMDF command definition file, `PMDF_COM:pmdf.cld`.

/MAXIMUM
/NOMAXIMUM (default)
The file PMDF_TABLE:maximum_command.dat is read when /MAXIMUM is specified. This file specifies near maximum table sizes but does not change any other command option file parameter settings. Only use this qualifier if the current table sizes are inadequate. The /NOIMAGE_FILE and /OPTION_FILE qualifiers should always be used in conjunction with this qualifier—it makes no sense to output the enormous command definition image that is produced by /MAXIMUM, but it does make sense to use /MAXIMUM to get past size restrictions in order to build a properly sized command option file so that a properly sized command definition image can be built with a subsequent CLBUILD invocation.

/OPTION_FILE[=file-spec]
/NOOPTION_FILE (default)
CLBUILD can optionally produce a command option file that contains correct table sizes to hold the command definitions which were just compiled (plus a little room for growth). The /OPTION_FILE qualifier causes this file to be read as input and a new such option file created as output. If /OPTION_FILE is specified with no value, then the file written will have the same name as the input command definition file, but with the file extension .cop; for instance, if the file PMDF_COM:pmdf.cld was the input parameter, then the default name for the output command option file would be PMDF_COM:pmdf.cop. If the /NOOPTION_FILE qualifier is specified (the default), then no option file will be output.

Note that use of the /MAXIMUM qualifier causes CLBUILD to read options from maximum_command.dat in addition to any command option file. This file specifies near maximum table sizes. Only use this qualifier if the current table sizes are inadequate, and only use it to create a new command option file. The /NOIMAGE qualifier should always be specified when /MAXIMUM is specified since a maximum-size image would be truly enormous and extremely wasteful.

/SIZES
/NOSIZES (default)
The /SIZES qualifier instructs PMDF CLBUILD to output information on the sizes of the uncompiled command definitions.

/STATISTICS
/NOSTATISTICS (default)
The /STATISTICS qualifier instructs CLBUILD to output information on the compiled conversion tables. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the /OPTION_FILE qualifier is needed.

EXAMPLES

The standard commands used to compile the basic PMDF command definition file and restart PMDF to use it, plus update the system DCL tables, are:

```
$ PMDF CLBUILD/OPTION_FILE/IMAGE_FILE=PMDF_COMMAND_DATA PMDF_COM:pmdf.cld
$ INSTALL REPLACE PMDF_COMMAND_DATA
$ SET COMMAND/TABLE=SYS$COMMON:[syslib]dcltables.exe -
_$  /OUTPUT=SYS$COMMON:[syslib]dcltables.exe PMDF_COM:pmdf.cld
$ INSTALL REPLACE SYS$LIBRARY:dcltables.exe
```

CNBUILD—Configuration compiler

Compile the PMDF configuration, alias, mapping, security, system wide filter, circuit check, and option files into an OpenVMS shareable image.

SYNTAX

PMDF CNBUILD

Command Qualifiers	Defaults
/IMAGE_FILE=file-spec	*/IMAGE_FILE=PMDF_CONFIG_DATA*
/MAXIMUM	*/NOMAXIMUM*
/OPTION_FILE=file-spec	*/OPTION_FILE=PMDF_OPTION_FILE*
/SIZES	*/NOSIZES*
/STATISTICS	*/NOSTATISTICS*

restrictions *None.*

PARAMETERS

None.

DESCRIPTION

The CNBUILD utility compiles the textual configuration, option, mapping, security, conversion, system wide filter, and alias files into a single OpenVMS shareable image. The resulting image, PMDF_CONFIG_DATA (usually PMDF_EXE:CONFIG_DATA.EXE), can then be installed with the OpenVMS INSTALL utility.

Whenever a component of PMDF (*e.g.,* a channel program) must read any possibly compiled configuration component, it first checks to see if the PMDF_CONFIG_DATA image exists. If it does, the image is merged into the running program using the OpenVMS RTL routine LIB$FIND_IMAGE_SYMBOL. There are five exceptions to this rule. The first is CNBUILD itself, which for obvious reasons always reads the text files and never tries to load the image form of the configuration data. The remaining four exceptions are TEST/REWRITE, and TEST/MAPPING, which can all be instructed with the /NOIMAGE_FILE qualifier to ignore any compiled image information. This facility in TEST/REWRITE is useful for testing changes prior to compiling them.

The reason for compiling configuration information is simple: performance. The only penalty paid for compilation is the need to rebuild and reinstall the file any time the configuration or alias files are edited. Also, be sure to restart any channels or components which load the configuration data only once when they start up (*e.g.,* the PMDF multithreaded TCP SMTP server, the POP or IMAP servers, FAX_RECEIVE, BITNET channels, or, if using PMDF-MR for MR TS replacement, the All-in-1 Sender, All-in-1 Fetcher, and MailWorks server).

Once you begin to use a compiled configuration, it will be necessary to recompile the configuration every time changes are made to any of the following files: the PMDF configuration file, pmdf.cnf (or any files referenced by it); the system alias file, aliases.; the system mapping file, mappings.; the PMDF option file, option.dat; the conversions file, conversions., the system wide filter file, pmdf.filter, the circuit check configuration file, circuitcheck.cnf, or the security configuration file, security.cnf. Until such time that the configuration is recompiled and reinstalled, changes to any of these files will not be visible to the running PMDF system.

See Chapter 8 for further details on the use of compiled configurations.

COMMAND QUALIFIERS

/IMAGE_FILE=file-spec
/NOIMAGE_FILE
By default, CNBUILD creates as output the file PMDF_CONFIG_DATA. With the /IMAGE_FILE qualifier, an alternate file name may be specified.

When the /NOIMAGE_FILE qualifier is specified, CNBUILD does not produce an output file. This qualifier is used in conjunction with the /OPTION_FILE qualifier to produce as output an option file which specifies table sizes adequate to hold the configuration required by the processed input files.

/MAXIMUM
/NOMAXIMUM (default)
The file PMDF_TABLE:maximum.dat is read in addition to PMDF_OPTION_FILE when /MAXIMUM is specified. This file specifies near maximum table sizes but does not change any other option file parameter settings. Only use this qualifier if the current table sizes are inadequate. The /NOIMAGE and /OPTION_FILE qualifiers should always be used in conjunction with this qualifier—it makes no sense to output the enormous configuration that is produced by /MAXIMUM, but it does make sense to use /MAXIMUM to get past size restrictions in order to build a properly sized option file so that a properly sized configuration can be built with a subsequent CNBUILD invocation.

/OPTION_FILE[=file-spec]
/NOOPTION_FILE (default)
CNBUILD can optionally produce an option file that contains correct table sizes to hold the configuration that was just compiled (plus a little room for growth). The /OPTION_FILE qualifier causes this file to be output. By default, this file is the file pointed to by the PMDF_OPTION_FILE logical, normally PMDF_TABLE:option.dat. The value on the /OPTION_FILE qualifier may be used to specify an alternate file name. If the /NOOPTION_FILE qualifier is given, then no option file will be output.

CNBUILD always reads any option file that is already present via the PMDF_OPTION_FILE logical name; use of this qualifier will not alter this behavior. However, use of the /MAXIMUM qualifier causes CNBUILD to read PMDF options from the PMDF_TABLE:maximum.dat in addition to reading PMDF_OPTION_FILE. This file specifies near maximum table sizes. Only use this qualifier if the current table sizes are inadequate, and only use it to create a new option file. The /NOIMAGE qualifier should always be specified when /MAXIMUM is specified since a maximum-size image would be truly enormous and extremely wasteful.

/SIZES
/NOSIZES (default)
The /SIZES qualifier instructs PMDF CNBUILD to output information on the sizes of the elements of the uncompiled configuration.

/STATISTICS
/NOSTATISTICS (default)
The /STATISTICS qualifier instructs CNBUILD to output information on how much of the various tables in the compiled configuration were actually used to store data. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the /OPTION_FILE qualifier is needed.

EXAMPLES

1
```
$ PMDF CNBUILD
$ INSTALL REPLACE PMDF_CONFIG_DATA
```

Above are the standard commands used to regenerate and reinstall a compiled configuration. After compiling the configuration, install it with the DCL INSTALL command and then restart any programs which may need to reload the new configuration. (For instance, it is necessary to restart the PMDF multithreaded TCP SMTP server with the "PMDF RESTART SMTP" command after recompiling the configuration.)

2
```
$ PMDF CNBUILD/NOIMAGE_FILE/OPTION_FILE/MAXIMUM
$ PMDF CNBUILD
$ INSTALL REPLACE PMDF_CONFIG_DATA
```

Use the sequence of three commands shown above when you encounter the infamous "No room in table" error message.

CONFIGURE—Create PMDF configuration files

Create basic PMDF configuration files.

SYNTAX

PMDF CONFIGURE *[product-or-component-name]*

restrictions

To write "live" files, must have write access to the PMDF table directory, `PMDF_TABLE:`.

PARAMETERS

product-or-component-name
The product name, *i.e.*, `mta`, `access`, `firewall`, `lan`, or the component name, *i.e.*, `dispatcher`, `mailbox_servers`, or `queues`. This parameter need not be specified if the product is PMDF or PMDF-MTA.

DESCRIPTION

CONFIGURE is an interactive command line utility for creating basic PMDF configuration files. Note that there is also a newer, web-based configuration utility for generating PMDF-MTA (including mailbox servers) configurations; see the *PMDF Installation Guide* for additional details.

Most fundamentally, it is used to generate a basic PMDF configuration file, alias file, mappings file, and security configuration file, usually `PMDF_TABLE:pmdf.cnf`, `PMDF_TABLE:aliases.`, `PMDF_TABLE:mappings.`, and `PMDF_TABLE:security.cnf`, respectively. The utility prompts for answers to questions regarding a site's node names and network connectivity, and then creates the basic files in accord with the answers to those questions.

The utility is also used to configure optional PMDF layered products, and to configure various PMDF components.

This utility is usually run when PMDF is first installed. It may also be convenient to run this utility, rather than manually editing the PMDF configuration file, after changes in a site's network configuration, such as the addition or removal of nodes, or a change in the status of a site's access to a larger network. Although by default this utility writes "live" files, overwriting any existing configuration and alias file, different file names may be specified, which can be useful for comparison or testing purposes. Note that since this utility does not take into account any pre-existing configuration file and alias file, any manual changes made to such files must be re-entered into the new files.

For a complete description and examples of using this utility to create configuration files for PMDF products or components, see the *PMDF Installation Guide, OpenVMS Edition*.

CONVERT—Convert a file

Convert a file.

SYNTAX

CONVERT *input-file output-file*

Command Qualifiers	Defaults
/CHECKSUM	*/NOCHECKSUM*
/DEBUG	*/NODEBUG*
/FTYPE=type	*See text*
/FCREATOR=creator	*See text*
/FDL=fdl-spec	*None*
/LINE_LENGTH=value	*/LINE_LENGTH=0*
/MPARAMETERS=param-list	*See text*
/MSUBTYPE=type	*See text*

Positional Qualifiers	Defaults
/CHARSET=charset	*None*
/ENCODING=encoding	*None*
/FILENAME=name	*/NOFILENAME*
/HEADER	*/NOHEADER*
/MODE=mode	*None*
/PARAMETERS=param-list	*None*
/SUBTYPE=subtype	*None*
/TYPE=type	*None*

restrictions *None.*

PARAMETERS

input-file[,...]
A comma separated list of one or more text files to be converted. All specified input files are read and merged into a single output file.

output-file
The name of the output file to write.

DESCRIPTION

CONVERT is a utility to convert files from one format to another.

COMMAND QUALIFIERS

/CHARSET=charset
Specify the character set for the part.

/CHECKSUM
/NOCHECKSUM (default)

/DEBUG
/NODEBUG (default)
Control whether or not to display debug output.

/ENCODING=encoding
Specify the encoding used for the part.

/FILENAME=name
/NOFILENAME (default)
Specify a file name to include in the MIME labelling.

/FTYPE=type

/FCREATOR=creator

/FDL=fdl-spec

/HEADER
/NOHEADER (default)
Specify whether MIME headers are present or should be written to the part. The default is `/NOHEADER`, MIME headers are neither looked for nor generated.

/LINE_LENGTH=value

/MODE=mode
Specify the conversion mode. The mode value may be any of `CRATTRIBUTE`, `LFATTRIBUTE`, `CRLFATTRIBUTE`, `BLOCK`, `RECORD`, `TEXT`, `POSTSCRIPT`, `ENRICHED`, `FLOWED`, `HTML`, `DOUBLEAPPLE`, `SINGLEAPPLE`, `BINHEX`, or `VIRUSSCAN`.

/MPARAMETERS=param-list

/MSUBTYPE=type

/PARAMETERS=param-list

/SUBTYPE=subtype
Specify the MIME subtype.

/TYPE=type
Specify the MIME type.

EXAMPLES

1 $ **PMDF CONVERT A.MACB/MODE=MACBINARY A.SINGLE/MODE=SINGLE**

The command above is an example of converting a file from Macbinary format to Applesingle format; that is, this is an example of extracting just the data fork from the Macbinary format file.

convert_cache.com—Perform a CONVERT/RECLAIM on the queue cache

Perform a CONVERT/RECLAIM operation on the queue cache database.

SYNTAX

@PMDF_COM:convert_cache.com

DESCRIPTION

The `convert_cache.com` utility performs a `CONVERT/RECLAIM` operation on the queue cache database. If you encounter difficulties with the queue cache database which a `CACHE/SYNCHRONIZE` command does not resolve, using this utility should be your next step.

EXAMPLES

To convert the queue cache database issue the command

```
$ @PMDF_COM:convert_cache.com
```

COUNTERS/CLEAR—Clear the in-memory counters

Clear the node-specific, in-memory cache of counters.

SYNTAX

COUNTERS/CLEAR

Command Qualifiers	Defaults
/ASSOCIATIONS	*/ASSOCIATIONS*
/CHANNELS	*/CHANNELS*

restrictions

WORLD privilege is required to in order to use this utility. If the in-memory section did not already exist (so that a new one must be created), then SYSGBL and PRMGBL privileges are also required. If a new cluster-wide, on-disk database must be created, then privileges sufficient to create a file in the PMDF_TABLE: directory are required.

PARAMETERS

None.

DESCRIPTION

The PMDF COUNTERS/CLEAR command is used to clear the values in the node-specific, in-memory section of counters. The command creates the node-specific, in-memory section of association and channel counters if it does not already exist. Then it zeros all fields in the in-memory section. Note that the counters will be zeroed without first merging their values into the cluster-wide database of channel counters. If a cluster-wide, on-disk database does not already exist, a new one will be created. Finally, the fields in the on-disk database for numbers of stored messages, message recipients, and message volumes are set based on the entries in the PMDF queue cache database.

Either the association counters, or channel counters, or both, may be cleared. The default is to clear both association and channel counters.

If you want to update the on-disk database with the old in-memory values before clearing them, then you should issue a

```
$ PMDF COUNTERS/SYNCHRONIZE
```

command before issuing the PMDF COUNTERS/CLEAR command.

You may also want to issue a

```
$ PMDF CACHE/SYNCHRONIZE
```

command before issuing the PMDF COUNTERS/CLEAR command, to ensure that the queue cache database values (which will be used to set some of the on-disk database values) are themselves current.

COMMAND QUALIFIERS

/ASSOCIATIONS (default)
/NOASSOCIATIONS
This qualifier specifies whether to clear the in-memory cache of association counters.

/CHANNELS (default)
/NOCHANNELS
This qualifier specifies whether to clear the in-memory cache of channel counters.

COUNTERS/CRDB—Create a cluster-wide counters database

Create a cluster-wide, on-disk database of association and channel counters.

SYNTAX

COUNTERS/CRDB

Command Qualifiers	Defaults
None.	*None.*

restrictions Requires sufficient privileges to create a file in the PMDF_TABLE: directory; if a in-memory section must also be created, SYSGBL and PRMGBL privileges are required.

PARAMETERS

None.

DESCRIPTION

A new, cluster-wide database of channel counters is created with the PMDF COUNTERS/CRDB command. The new database will have all counters zeroed except for the counts of stored messages, recipients, and message volumes for each channel. Those counts will be determined by the entries in the PMDF queue cache database. In addition, if an in-memory section for association and channel counters on this node does not already exist, it will be created as well.

Once an on-disk database exists, its values may be updated from the node-specific, in-memory sections by using the PMDF COUNTERS/SYNCRONIZE command.

Note that since some initial database values will be set based on entries in the PMDF queue cache database, you may want to issue a

```
$ PMDF CACHE/SYNCHRONIZE
```

command before issuing the PMDF COUNTERS/CRDB command, to ensure that the queue cache database values are themselves current.

COUNTERS/SHOW—Display the counters

Display the contents of the cluster-wide database of counters.

SYNTAX

COUNTERS/SHOW

Command Qualifiers	Defaults
/ASSOCIATIONS	*/ASSOCIATIONS*
/CHANNELS	*/CHANNELS*
/HEADERS	*/HEADERS*
/OUTPUT=file-spec	*None*
/TODAY	*/TODAY*

restrictions

Normally WORLD privilege is all that is required. But if the cluster-wide, on-disk database must be created, then privileges sufficient to create a file in the PMDF_TABLE: directory are required; or if the node-specific, in-memory section must be created, then SYSGBL and PRMGBL privileges are required.

DESCRIPTION

The contents of the cluster-wide association and channel counters database may be displayed with the PMDF COUNTERS/SHOW command. A PMDF COUNTER/SYNCHRONIZE command is implicitly performed by this command; the database contents are synchronized with the in-memory section(s) before being displayed.

Note that as part of the implicit PMDFCOUNTERS/SYNCHRONIZE operation, if the cluster-wide, on-disk database does not already exist, the PMDF COUNTERS/SHOW command will create it. And if the node-specific, in-memory cache of counters does not already exist, the PMDF COUNTERS/SHOW command will create it too.

COMMAND QUALIFIERS

/ASSOCIATIONS (default)
/NOASSOCIATIONS
This qualifier specifies whether to show the in-memory cache of association counters.

/CHANNELS (default)
/NOCHANNELS
This qualifier specifies whether to show the in-memory cache of channel counters.

/HEADERS (default)
/NOHEADERS
Controls whether or not a header line describing each column in the table of counters is output.

/OUTPUT=file-spec
Direct the output to the specified file. By default the output appears on your display.

/TODAY (default)
/NOTODAY
This qualifier specifies whether to show PMDF's count for the number of messages processed this day. Note that as discussed in Section 31.4.1, PMDF counters are intentionally designed to be lightweight and as such by design, the value shown becomes increasingly likely to be an undercount as message volume increases. So high volume sites (sites with an unlimited volume PMDF license) in particular should not place too much credence in the reported number.

EXAMPLES

To display the counters for all channels and associations, issue the command

```
$ PMDF COUNTERS/SHOW
4263 messages processed so far today
30000 messages per day are permitted by your license

Channel                      Messages  Recipients      Blocks
-------------------------  ----------  ----------  ----------
l
    Received                     3863        3881       25786
    Stored                         89          89         460
    Delivered                    3876        3894       26018 (3859 first time)
    Submitted                      99         114        1611
    Attempted                      17          17          25
    Rejected                        0           0           0
    Failed                          1           1           6

    Queue time/count        29794837/3877 = 7.68502E3
    Queue first time/count  18904343/3860 = 4.8975E3

tcp_local
    Received                      208         217        4153
    Stored                          3           3           9
    Delivered                     200         212        2461 (197 first time)
    Submitted                    4053        4078       25919
    Attempted                       7           7           0
    Rejected                       46          68           0
    Failed                         14          14        1695

    Queue time/count        1106266/211 = 5.24297E3
    Queue first time/count  455897/208 = 2.19181E3

    Current In Assocs            127
    Total In Assocs             1056
    Total Out Assocs             132
    Rejected Out Assocs           11
    Failed Out Assocs              1
```

```
Channel      Timestamp    Association
------------ ------------ ------------------------------------------------
tcp_local    01-Feb 00:27 TCP|192.160.253.70|25|192.160.253.66|3465
tcp_local    25-Jan 00:31 TCP|192.160.253.70|5|192.160.253.66|3496
tcp_local    26-Jan 14:50 TCP|192.160.253.70|25|192.160.253.66|2086
tcp_local    05-Feb 12:23 TCP|192.160.253.70|25|192.160.253.66|3593
tcp_local    01-Feb 00:34 TCP|192.160.253.70|25|192.160.253.66|3581

...
$
```

COUNTERS/SYNCHRONIZE—Synchronize in-memory counters with the cluster-wide database

Synchronize each of the node-specific, in-memory caches of channel counters with the cluster-wide database.

SYNTAX

COUNTERS/SYNCHRONIZE

Command Qualifiers	Defaults
None.	*None.*

restrictions

Normally, just WORLD privilege is required to use this utility. However, if the node-specific, in-memory section must be created, then SYSGBL and PRMGBL are required; or if the cluster-wide, on-disk database must be created, then privileges sufficient to create a file in the `PMDF_TABLE:` directory are required.

PARAMETERS

None.

DESCRIPTION

To synchronize each of the node-specific, in-memory caches of channel counters with the cluster-wide database, issue a `PMDF COUNTERS/SYNCHRONIZE` command. The command will not return control back to you until all the caches have been synchronized. The `PMDF COUNTERS/SYNCHRONIZE` command signals each PMDF counters synchronization process in the cluster—there should be one such process on each node running PMDF. Note that on each node, the synchronization can only be performed if the PMDF counters synchronization process is running on that node.

Assuming that the PMDF counters synchronization process is running on each node, then for each node the node-specific, in-memory cache will be created, if it does not already exist. If the cluster-wide, on-disk database does not exist, it will be created. The in-memory cache values will be used to update the on-disk database, and then the on-disk database values for stored messages, recipients, and volume will be set by scanning the PMDF queue cache database.

COUNTERS/TODAY—Display number of messages processed today

Display PMDF's count of the number of messages processed so far today.

SYNTAX

COUNTERS/TODAY

Command Qualifiers	Defaults
None.	*None.*

restrictions WORLD privilege is required.

DESCRIPTION

PMDF's count of the number of messages processed so far today may be displayed with the `PMDF COUNTERS/TODAY` command.

Note that as discussed in Section 31.4.1, PMDF counters are intentionally designed to be lightweight and as such by design, the value shown becomes increasingly likely to be an undercount as message volume increases. So high volume sites (sites with an unlimited volume PMDF license) in particular should not place too much credence in the reported number.

EXAMPLES

To display PMDF's count of the number of messages processed today, issue the command

```
$ PMDF COUNTERS/TODAY
4263 messages processed so far today
30000 messages per day are permitted by your license
$
```

CRDB—Create database

CRDB is a utility used to create and update PMDF database files.

SYNTAX

PMDF CRDB *input-file-spec[,...] output-database-spec*

Command Qualifiers	Defaults
/APPEND	*/NOAPPEND*
/COUNT	*/COUNT*
/DELETE	*/NODELETE*
/DUMP	*See text*
/DUPLICATES	*/NODUPLICATES*
/EXCEPTION_FILE=file-spec	*/NOEXCEPTION_FILE*
/EXCLUDE=(suffix1[,...])	*See text*
/FAST_LOAD	*/FAST_LOAD*
/HUGE_RECORDS	*/NOHUGE_RECORDS*
/LONG_RECORDS	*/NOLONG_RECORDS*
/QUOTED	*/NOQUOTED*
/REMOVE	*/NOREMOVE*
/SCRATCH_DISK=device-spec	*See text*
/STATISTICS	*/STATISTICS*
/STRIP_COLONS	*/NOSTRIP_COLONS*
/WRITE_CHECK	*/NOWRITE_CHECK*

restrictions

None.

prompts

Input file: *input-file-spec[,...]*
Output database: *output-database-spec*

PARAMETERS

input-file-spec[,...]
A comma separated list of one or more text files containing the entries to be placed into the database. Each line of the text files must correspond to a single entry. All specified input files are read and merged into a single output database.

output-database-spec
The name of the database file to write the database to. This may be a new or existing database. If the `/NOFAST_LOAD` qualifier is specified and the database already exists no new database will be created; records will simply be added to the existing database.

DESCRIPTION

CRDB is a utility to create and or update PMDF database files. CRDB simply converts a plain text file into PMDF database records and either builds a new database or updates the records in an existing database.

In general, each line of the input file must consist of a left hand side and a right hand side. The two sides are separated by one or more spaces or tabs. The left hand side is limited to 32 characters in a short database (the default variety), 80 characters in a long database, or 252 characters in a huge database. The right hand side is limited to 80 characters in a short database, 256 characters in a long database, or 1024 characters in a huge database. Spaces and tabs may not appear in the left hand side (but see the description of the /QUOTED qualifier below).

The format of the input files is described in the sections describing each particular PMDF database. For instance, the format of the input files for an alias database is described in Section 3.1.2; the format of the input files for the domain database (rewrite rule database) is described in Section 2.2.9; the format of the input files for the address reversal database is described in Section 3.3.2.

COMMAND QUALIFIERS

/APPEND
/NOAPPEND (default)
If /APPEND is specified, the database is loaded with RMS $PUT operations. If the database already exists it will be appended to; if not, a new database will be created. Duplicate keys (left hand sides) will replace existing entries in databases created with /NODUPLICATE, so the last occurrence of a given key will be the one that is used.

/NOAPPEND is a synonym for for /FAST_LOAD.

/COUNT (default)
/NOCOUNT
Controls whether or not a count is output after each group of 100 input lines are processed. This qualifier only applies if /APPEND is in effect; it is ignored in /FAST_LOAD mode.

/DELETE
/NODELETE (default)
If /DELETE is specified, the given entries are deleted from the database. The input file should contain one key value per line for the entries to delete. The data portion of the line is ignored. If the database was created with /DUPLICATE, for multiple entries with the same key value, only the first entry is deleted.

/DUMP
PMDF CRDB/DUMP is a synonym for PMDF DUMPDB. It is used to cause PMDF CRDB to dump an existing database to a flat text file—or to SYS$OUTPUT if no output file is specified. When /DUMP is specified, the parameters to PMDF CRDB are interpreted as the input database specification, and optionally a flat text file to which to write the output. No other qualifiers are valid when /DUMP is specified.

/DUPLICATES
/NODUPLICATES (default)
Controls whether or not duplicate records are allowed in the output file. Currently duplicate records are of use only in the domain databases (rewrite rule databases) and databases associated with the directory channel.

/EXCEPTION_FILE=file-spec
/NOEXCEPTION_FILE
CRDB may encounter records that cannot be loaded into the database. This usually means that in /FAST_LOAD mode these records had keys (left hand sides) that were duplicates of other keys previously encountered in the input file. When /FAST_LOAD is used (the default), these exception records can optionally be written to a separate output file for later examination. The /EXCEPTIONS_FILE qualifier controls the writing of this file. Note that the lines in this file are not plain text; they are formatted as database entries.

/EXCLUDE=(suffix[,...])
Any left-hand side entries ending with the string **suffix** will be excluded from the database. By default no entries are omitted.

/FAST_LOAD (default)
/NOFAST_LOAD
This qualifier controls whether or not the fast load algorithm is used. If /FAST_LOAD is specified, callable CONVERT is used to build the database. A new database is always created. If records with duplicate keys (left hand sides) are encountered in the input stream and /NODUPLICATE is in effect, the specific occurrence that will be used is unpredictable.

/NOFAST_LOAD is a synonym for for /APPEND.

/LONG_RECORDS
/NOLONG_RECORDS (default)
/HUGE_RECORDS
/NOHUGE_RECORDS
These qualifiers control the size of the output records. By default left hand sides are limited to 32 characters and right hand sides are limited to 80 characters. If /LONG_RECORDS is specified the limits are changed to 80 and 256, respectively. If /HUGE_RECORDS is specified the limits are changed to 252 and 1024 characters, respectively. Currently, /HUGE_RECORDS databases are supported only for the alias database.

/QUOTED
/NOQUOTED (default)
This qualifier controls the handling of quotes. Normally CRDB pays no particular attention to double quotes. If /QUOTED is specified, CRDB matches up double quotes in the process of determining the break between the left and right hand sides of each input line. Spaces and tabs are then allowed in the left hand side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of the database keys. Note: The quotes are not removed unless the /REMOVE qualifier is also specified.

/REMOVE
/NOREMOVE (default)
This qualifier controls the removal of quotes. If CRDB is instructed to pay attention to quotes, the quotes are normally retained. If /REMOVE is specified, CRDB removes the outermost set of quotes from the left hand side of each input line. Spaces and

tabs are then allowed in the left hand side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of the database keys. Note: /REMOVE is ignored if /QUOTED is not in effect.

/SCRATCH_DISK=device-spec

CRDB uses one or two temporary scratch files to build the database if /FAST_LOAD is used (the default). The /SCRATCH_DISK qualifier can be used to specify the device on which these files are created. It may be useful to place these files in a specific place, either to improve performance or to get around protection and quota limitations.

If /SCRATCH_DISK is not specified, the temporary files will be created on the disk specified by PMDF_SCRATCH. If PMDF_SCRATCH is not defined, the temporary files will be created on the disk specified by SYS$SCRATCH. If SYS$SCRATCH is not defined, the temporary files will be created on whatever disk the current default directory is on.

/STATISTICS (default)
/NOSTATISTICS

Controls whether or not some simple statistics are output by CRDB, including number of entries (lines) converted, number of exceptions (usually duplicate records) detected, and number of entries that could not be converted because they were too long to fit in the output database. /NOSTATISTICS suppresses output of this information.

/STRIP_COLONS
/NOSTRIP_COLONS (default)

The /STRIP_COLONS qualifier instructs CRDB to strip a trailing colon from the right end of the left hand side of each line it reads from the input file. This is useful for turning former alias file entries into an alias database.

/WRITE_CHECK
/NOWRITE_CHECK (default)

Controls whether or not RMS write checking is enabled on the output database. This only applies to /FAST_LOAD mode; this qualifier is ignored otherwise.

EXAMPLES

The following commands may be used to create an alias database with "long" record entries; note that the creation is performed in a two-step process using a temporary database to minimize any window of time, such as during database generation, when the database would be locked and inaccessible to PMDF:

```
$ PMDF CRDB/LONG_RECORDS PMDF_TABLE:aliases.txt aliases.tmp
$ RENAME aliases.tmp PMDF_ALIAS_DATABASE
```

DCF—Document format converter

Convert WPS and DX files to other formats.

SYNTAX

PMDF DCF *input-file-spec output-file-spec*

Positional Qualifiers	Defaults
/FORMAT=type	*See text*

restrictions

- This utility is available only for OpenVMS VAX. and is only intended for internal use by the PMDF–MR channels.
- The file extensions of the input and output files are not user selectable.

prompts

Input file: *input-file-spec*
Output file: *output-file-spec*

PARAMETERS

input-file-spec
The name of an input file to convert. The format of the file must be specified with a positional /FORMAT qualifier. The file extension of the input file must match the specified format (see the /FORMAT qualifier description for details).

output-file-spec
The name of an output file to create. The format of the output file is selected with the /FORMAT positional qualifier. The file extension of the output file must match the specified format (see the /FORMAT qualifier description for details).

DESCRIPTION

The DCF utility is used by PMDF–MR to convert WPS and DX message bodyparts to ASCII. This facility is built from document conversion software supplied as part of the MRGATE kit (version 3.1 or 3.2), the Message Router VMS MAIL gateway. PMDF–MR will function without this utility but it will be unable to convert WPS and DX bodyparts to ASCII.

POSITIONAL QUALIFIERS

/FORMAT=type

This positional qualifier must be given for both the input and output file specifications. The allowable types are `ASCII`, `DX`, and `WPSFILE`. The file extension of the input and output file must be given and they must agree with the selected format: for ASCII files the file extension must be `.txt`, for DX files the file extension must be `.dx`, and for WPSFILE files the extension must be `.wpl`. Failure to use the proper extension will result in unpredictable results.

EXAMPLES

The following command may be used to convert a WPS-PLUS file to an ordinary text format:

```
$ PMDF DCF koala.wpl/FORMAT=WPSFILE mrsdd.txt/FORMAT=ASCII
```

All special text attributes (*e.g.,* bold, underlined, *etc.*), are lost in the conversion from WPS-PLUS to ASCII.

DUMPDB—Dump contents of a PMDF CRDB database to a file

Dump contents of a PMDF CRDB database to a file.

SYNTAX

PMDF DUMPDB *input-database-spec [output-file-spec]*

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

prompts
Input database: *input-database-spec[,...]*
Output file: *output-file-spec*

PARAMETERS

input-database-spec
The name of the database from which to read entries.

output-file-spec
The name of the ASCII file to which to write the entries stored in the database. If no output file is specified, the output is written to SYS$OUTPUT.

DESCRIPTION

The PMDF DUMPDB utility writes the entries in a PMDF CRDB database to a flat ASCII file.

EXAMPLES

The following command illustrates dumping the PMDF alias database to SYS$OUTPUT.

```
$ PMDF DUMPDB PMDF_ALIAS_DATABASE
!!PMDF CRDB/QUOTED/REMOVE/NOLONG_RECORDS  PMDF_ALIAS_DATABASE
adam.smith asmith@example.com
bob.brown bbrown@example.com
```

G3—Analyze G3 file

Analyze the contents of a PMDF–FAX G3 file and generate a DDIF file containing the G3 file's bitmaps.

SYNTAX

PMDF G3 *G3-file-spec [DDIF-file-spec]*

Command Qualifiers	Defaults
/CDA	*/CDA*
/EDIT	*None*
/INFORMATION	*/INFORMATION*
/MAGNIFICATION=factor	*/MAGNIFICATION=1*

restrictions

- Will only operate on PMDF–FAX G3 files; files with other formats will be rejected.
- Requires SYSLCK privilege to operate.
- This utility is supplied only with the PMDF-FAX optional layered product.

prompts

Input file: *G3-file-spec*
Output file: *DDIF-file-spec*

PARAMETERS

G3-file-spec
The name of a PMDF–FAX G3 file. G3 will check that a proper G3 file is specified and will abort execution when supplied the name of a non-G3 file. Only a single input file name may be supplied; wildcards are not allowed.

DDIF-file-spec
The name of a DDIF file (CDA document) to which to write the input file's bitmaps. This file will be created by G3 and may subsequently be viewed with the CDA viewer as shown in the example below. If the /NOCDA or /EDIT qualifies are used, then this parameter must be omitted.

DESCRIPTION

The G3 utility may be used to analyze the contents of a G3 file in a G3_TO_FAX channel queue directory. In addition, the G3 utility is capable of converting a G3 file to a DDIF file which may then be viewed with the HP CDA Viewer. If you merely want to view header and delivery information in a G3 file, then use the /NOCDA qualifier. This will disable the time-consuming output of a DDIF file containing the bitmap images stored in the G3 file.

Information about each of the To: recipients in a G3 file may be edited using the edit mode invoked with the /EDIT qualifier. This allows incorrect FAX telephone numbers to be edited, delivery to specific recipients to be cancelled, *etc.*

Note: The G3 utility is part of the PMDF-FAX software product. It is not part of the base PMDF distribution.

COMMAND QUALIFIERS

/CDA (default)
/NOCDA
By default a CDA document is output by G3. This document is a DDIF file containing the individual bitmaps stored in the G3 file. When the /CDA qualifier is used, the *DDIF-file-spec* parameter must be supplied.

The *DDIF-file-spec* parameter must be omitted when the /NOCDA qualifier is specified. The /MAGNIFICATION qualifiers may not be used in conjunction with the /NOCDA qualifier.

/EDIT
When the /EDIT qualifier is given, the G3 utility operates in edit mode, presenting each To: address contained in the G3 file for editing. The /CDA, /MAGNIFICATION, and /INFORMATION qualifiers may not be used in conjunction with the /EDIT qualifier. In addition, the output DDIF file specification may not be given when this qualifier is used.

/INFORMATION (default)
/NOINFORMATION
By default, information about the G3 file is displayed (*e.g.,* the recipients of the FAX message embodied by the G3 file and the delivery status associated with each recipient). When the /NOINFORMATION qualifier is used, the display of this information is suppressed.

/MAGNIFICATION=factor
When a CDA document is output, it is scaled to be the same size as the actual FAX message (approximately 8.5 by 10.5 inches). This corresponds to a magnification of 1 (a scaling by a factor of unity). With the /MAGNIFICATION qualifier, the document may be scaled upwards or downwards by a prescribed scaling factor. For instance, to reduce the document to half size, use the qualifier /MAGNIFICATION=0.5. This qualifier may not be used in conjunction with the /NOCDA qualifier.

EXAMPLES

The following example illustrates the use of the CDA Viewer to view the FAX images contained in a G3 file.

```
$ PMDF G3 PMDF_QUEUE:[g3_to_fax]xyzzy.01 fax.ddif
$ VIEWER/INTERFACE=DECWINDOWS/OVERRIDE fax.ddif
```

INSTALL—Install PMDF images

Install or deinstall PMDF images and databases.

SYNTAX

PMDF INSTALL *operation-type*

Command Qualifiers	Defaults
None.	*None.*

restrictions Requires CMKRNL privilege.

prompts Operation: *operation-type*

PARAMETERS

operation-type
The type of operation to perform. May be one of ADD, CREATE, DELETE, LIST, REMOVE, or REPLACE.

DESCRIPTION

The PMDF INSTALL utility is used to install or deinstall PMDF images. This utility invokes the command file `image_install.com` with a command of the form

```
$ @PMDF_COM:image_install.com PMDF operation-type
```

where *operation-type* is one of ADD, CREATE, DELETE, LIST, REMOVE, or REPLACE. Each of these operations corresponds to the OpenVMS INSTALL utility operation of the same name; consult the *OpenVMS Install Utility Reference Manual* for descriptions of these operations. `image_install.com` invokes the OpenVMS INSTALL utility and executes the operation *operation-type* for each file specified in the the file `pmdfimage.dat` in the `PMDF_COM:` directory. (Exclamation marks are used in that file to introduce comments.)

The `pmdfimage.dat` file is reserved for PMDF use and should not be modified. PMDF INSTALL will also use an optional, site-supplied `PMDF_COM:siteimage.dat` file, of the same format as the `pmdfimage.dat` file, listing additional site-specific files. Thus sites that want to install additional, site-specific PMDF images, should do so by adding them to their own `siteimage.dat` file.

Note that the PMDF startup procedure installs PMDF at system startup by issuing the command

```
$ @PMDF_COM:image_install.com PMDF CREATE
```

The PMDF INSTALL command is not used by the startup procedure since not all sites insert the PMDF verb into their system DCL tables.

CMKRNL privilege is required to use the PMDF INSTALL utility.

EXAMPLES

The following command may be used to deinstall any installed PMDF images:

```
$ PMDF INSTALL DELETE
```

KILL—Kill all PMDF component processes

Kill all PMDF component processes.

SYNTAX

PMDF KILL

Command Qualifiers	Defaults
None.	*None.*

restrictions

Must have privileges sufficient to perform a STOP/ID upon the processes in question.

PARAMETERS

None.

DESCRIPTION

The PMDF KILL utility prompts you for each PMDF process and asks if you want to kill it.

The PMDF KILL utility immediately and indiscriminately kills the specified process (using STOP/ID), even if that process is in the middle of transferring e-mail. So use of the PMDF SHUTDOWN utility, which performs an orderly shutdown, is generally preferable.

LICENSE—Activate or deactivate PMDF bundle licenses

Activate or deactivate PMDF bundle licenses on a node.

SYNTAX

PMDF LICENSE *operation-type*

Command Qualifiers	Defaults
None.	*None.*

restrictions Requires SYSNAM and CMEXEC privileges.

prompts Operation: *operation-type*

PARAMETERS

operation-type
The type of operation to perform. May be one of LOAD or UNLOAD.

DESCRIPTION

The PMDF LICENSE utility is used to activate (load) or deactivate (unload) a PMDF license Product Authorization Key (PAK) on a given OpenVMS system. The PMDF license PAK must already be loaded with the OpenVMS LICENSE utility. However, that is not sufficient to activate the license. PMDF's model for debitting license usage units (based upon CPU type) differs from HP's. Because of this difference, an additional utility—the PMDF LICENSE utility – is required to properly debit usage units from a PMDF license PAK.

Normally the PMDF LICENSE utility is run at system startup by the `pmdf_startup.com` command procedure. However, it may also be run manually. When it is used to unload a license, the usage units used by that machine are made available for use by other systems in the cluster.

Note that the use of the OpenVMS LICENSE utility is sufficient for loading and activating license PAKs for PMDF-MTA, PMDF-MSGSTORE, PMDF-POPSTORE, and PMDF-TLS. The PMDF LICENSE utility does not need to be (and cannot be) used to activate license PAKs for PMDF-MTA or the listed layered products.

EXAMPLES

To activate a PMDF license, use the command

```
$ PMDF LICENSE LOAD
%PMDF-I-LOADED, 1 license unit loaded
```

Had the license already of been activated then the informational message

```
%PMDF-I-ALREADY, 1 license unit already loaded
```

would have been displayed. To deactivate a license, use the command

```
$ PMDF LICENSE UNLOAD
%PMDF-I-UNLOADED, 1 license unit unloaded
```

PASSWORD—Set remote authentication password

Set password for remote authentication, *e.g.*, POP client (APOP), IMAP client (CRAM), or mailbox filter authentication.

SYNTAX

PMDF PASSWORD *[password]*

Command Qualifiers	Defaults
/CONVERT	*/CREATE*
/CREATE	*/CREATE*
/DELETE	*/CREATE*
/SERVICE=keyword	*/SERVICE=DEFAULT*
/SHOW	*/CREATE*
/TEST	*/CREATE*
/USER=username	*See text*

restrictions

All operations other than setting or verifying one's own password, or showing one's own password database entries, require privileges.

prompts

New password: *password*

PARAMETERS

password
The password to set. Note that APOP passwords are case sensitive.

DESCRIPTION

The PMDF PASSWORD utility is used to create and modify PMDF password database entries. This database may be used by POP clients issuing the APOP command, by IMAP clients using the CRAM-MD5 authentication mechanism, or possibly by users authenticating themselves to modify their personal mailbox filters.

Note that in general, just which source of password authentication information is used—whether the PMDF password database, or some other source—is controlled by the PMDF security configuration file; see That is, a connection comes in (POP, IMAP, or mailbox filtering) and is mapped to a security rule set; the security rule set in the PMDF security configuration then controls where and how authentication is performed for that connection.

For instance, the DEFAULT security rule set in PMDF's implicit security configuration (which applies if no security configuration file exists) checks first for a PMDF user profile password (PMDF MessageStore or PMDF popstore profile password), next for a PMDF password database entry, and finally falls through to checking for a system password entry.

Note that APOP and CRAM-MD5 passwords cannot be stored in the system password file. Therefore, in order to support use of the POP protocol's APOP command or AUTH command with CRAM-MD5, or the IMAP protocol's authenticate command with CRAM-MD5, the user must have a password entry stored in an authentication source other than (or in addition to) the system password file. The PMDF password database can be that additional authentication source.

Thus for instance, for a POP or IMAP connection handled by the DEFAULT security rule set, a user must either be a PMDF MessageStore user or a PMDF popstore user (in which case their PMDF user profile password is normally[1] sufficient for remote authentication), or if they are a legacy message store (VMS MAIL) user then they must have a PMDF password database entry in addition to their system password file entry.

For mailbox filter connections handled by the DEFAULT security rule set of PMDF's implicit security configuration, authentication will be performed preferentially against the PMDF user profile, if the user has a PMDF user profile entry (that is, a PMDF MessageStore or PMDF popstore profile entry), if not then against the PMDF password database, if the user has an entry in it, and finally, only if the user has neither sort of entry, against the system password file.

The above discussion regards whether the PMDF password database will actually be used as the source of authentication information. When the PMDF password database is used as the source of authentication information, then an additional issue can arise, namely which of a user's possibly multiple entries will be checked for the authentication. That is, a user can have multiple entries in the PMDF password database, one for each allowed /SERVICE value. The sort of connection (assuming that the PMDF password database is even checked) will control which /SERVICE entry is preferentially checked. Note that the sort of /SERVICE checked has nothing to do with the PMDF security configuration (which instead controlled whether or not the PMDF password database was queried at all); the sort of /SERVICE entry checked when the PMDF password database is queried has entirely to do with which component of PMDF is doing the querying (what sort of connection this regards).

Queries by the POP server will first check a user's /SERVICE=POP entry, but if such an entry does not exist will fall through to the user's /SERVICE=DEFAULT entry. Queries by the IMAP server will first check a user's /SERVICE=IMAP entry, but if such an entry does not exist will fall through to the user's /SERVICE=DEFAULT entry.

[1] The PMDF MessageStore and PMDF popstore, however, have a PWD_ELSEWHERE flag to say that their passwords are stored elsewhere; if this is set, even a PMDF MessageStore user or a PMDF popstore user might use a PMDF password database entry.

Queries for mailbox filtering will check which channel a user matches. For a user matching a msgstore channel, the mailbox filter query will preferentially use the user's /SERVICE=IMAP entry, but if such an entry does not exist will fall through to the user's /SERVICE=DEFAULT entry. For a user matching a popstore channel, the mailbox filter query will preferentially use the user's /SERVICE=POP entry, but if such an entry does not exist will fall through to the user's /SERVICE=DEFAULT entry. For a user matching the local channel, the mailbox filter query will use the user's /SERVICE=DEFAULT entry.

Most sites and users will not want to use /SERVICE specific password database entries. Then each user has one entry, their /SERVICE=DEFAULT entry, used whenever the PMDF password database is queried.

But for sites and users who do want to use /SERVICE specific password database entries, while the above description of /SERVICE specific probes may sound complicated, the goal is simply to query the "natural" password entry for each case.

COMMAND QUALIFIERS

/CONVERT
The format of the PMDF password database changed in PMDF V5.1 from that used previously. This qualifier is used to convert a PMDF V5.0 password database to the PMDF V5.1 and later format.

/CREATE
Create a PMDF password database entry. This qualifier is the default.

/DELETE
Delete a user/password entry pair from the PMDF password database.

/SERVICE=keyword
Specify for what service a particular password method and password value apply. The default service keyword is DEFAULT; POP and IMAP are other possible keywords.

/SHOW
Show a user/service/password-method entry in the PMDF password database. Note that this commmand does not show the password value.

/TEST
Compare a specified password against a password stored in the PMDF password database.

/USER=username
Set or show a password entry in the PMDF password database for the specified user. To show all users' entries specify the asterisk as a value.

EXAMPLES

To add a user JSMITH with password SeCrEt to the database, use the command

```
$ PMDF PASSWORD/USER=JSMITH "SeCrEt"
```

The user JSMITH may change his own password, with prompting so that the password is not printed on the screen, using the command

```
$ PMDF PASSWORD
Password:
```

To list all usernames that have an entry in the PMDF password database, use the following command:

```
$ PMDF PASSWORD/SHOW/USER=*
```

ERROR MESSAGES

`%PMDF-E-CANOPNPASS, Password file does not exist or cannot be opened`

The PMDF password database does not exist, or could not be opened.

`%SYSTEM-F-NOWORLD, operation requires WORLD privilege`

Must have WORLD privilege to use the PMDF PASSWORD/CONVERT command, or to specify an entry for a user other than oneself.

PROCESS—Show currently executing PMDF jobs

List currently executing PMDF jobs.

SYNTAX

PMDF PROCESS *[node]*

Command Qualifiers	Defaults
/MEMORY	*See text*

restrictions Requires WORLD privilege in order to see all processes.

PARAMETERS

node
The name of the node whose PMDF processes are to be displayed. The asterisk character, *, may be specified to display processes on all nodes in the cluster.

DESCRIPTION

Show current PMDF processes. Normally, the PMDF Service Dispatcher should always be present; additional processes may be present if messages are currently being processed, or if certain additional PMDF components are in use.

COMMAND QUALIFIERS

/MEMORY
Show the amount of memory being used by the processes.

EXAMPLES

The following command shows current PMDF processes:

```
$ PMDF PROCESS
VAX OpenVMS V6.2     on node NAPLES 15-NOV-2012 10:56:23.25
Physical memory 80 MB (163840 pages) up since 31-OCT-2012 06:34:27.50

----- The following are on node NAPLES, a VAXstation 4000-90A -----
  Pid    Process Name    State  Pri      I/O       CPU       Page flts  Pages
22600127 PMDF counters   HIB      8     2028   0 00:00:07.66      4390    203
22600372 <HTTP-01>       HIB      6     1824   0 00:00:02.74      4921    124
226002B2 <DISPATCHER-01> HIB      6     4412   0 00:00:10.04     15578    554
226002B3 <SMTP-01>       HIB      6     5249   0 00:00:28.85     22094   1246
226002B6 <POPPASSD-01>   HIB      6      166   0 00:00:01.62      5248    198
```

QCLEAN—Hold or delete matching messages from the PMDF queue area

Hold or delete message files from the PMDF queue area that contain specified substrings in their envelope From: address, Subject: header, or message content.

SYNTAX

PMDF QCLEAN *[channel]*

Command Qualifiers	Defaults
/CONTENT=substring	*None*
/DATABASE	*/DATABASE*
/DELETE	*/HOLD*
/DIRECTORY_TREE	*/DATABASE*
/ENV_FROM=substring	*None*
/HOLD	*/HOLD*
/MATCH=keyword	*/MATCH=AND*
/MIN_LENGTH=n	*/MIN_LENGTH=24*
/SUBJECT=substring	*None*
/THREADS=n	*/NOTHREADS*
/VERBOSE	*/NOVERBOSE*

restrictions

Privileges sufficient to read and delete files in the PMDF channel queue directory tree, as well as read and update the PMDF queue cache database, are required.

PARAMETERS

channel

Optional parameter which specifies a specific PMDF channel area to be searched for matching messages. `*` or `?` wildcard characters may be used in the channel specification.

DESCRIPTION

Hold or delete message files containing specific substrings in their envelope From: address, Subject: line, or content. By default, message files are held (/HOLD). Specify /DELETE to instead delete matching message files. The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers are used to specify the substrings for which to search.

Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used. The /MATCH qualifier controls whether a message file must contain all (/MATCH=AND, the default) or only one of (/MATCH=OR) the specified substrings in order to be held or deleted. The default is /MATCH=AND.

By default, each substring to be searched for must be at least 24 bytes long (/MIN_LENGTH=24). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the /MIN_LENGTH qualifier to override this limit. Also by default, only message files identified by the queue cache database are searched (/DATABASE). Use the /DIRECTORY_TREE qualifier to instead search all message files actually present in the channel queue directory tree.

The optional channel parameter restricts the search to message files in the specified channel. The channel parameter may use * and ? wild cards.

The /THREADS qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searchingg threads, specify /THREADS=*n*. The value *n* must be in the range 1-8. The default is /NOTHREADS.

COMMAND QUALIFIERS

/CONTENT=substring
/ENV_FROM=substring
/SUBJECT=substring
The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers are used to specify the substrings for which to search. Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used. When a combination of such qualifiers is used, the /MATCH qualifier controls whether the qualifiers are interpreted as further restrictions (/MATCH=AND), or as alternatives (/MATCH=OR).

/DATABASE (default)
/DIRECTORY_TREE
The /DATABASE qualifier, the default, specifies that only message files identified by the queue cache database be searched. Use the /DIRECTORY_TREE qualifier to instead search all message files actually present in the channel queue directory tree.

/DELETE
/HOLD (default)
/HOLD is the default and means that matching message files will be held. Specify /DELETE to instead delete matching message files.

/MATCH=keyword
The default is /MATCH=AND, meaning that any criteria specified by /CONTENT, /ENV_FROM, and /SUBJECT qualifiers must all match in order for the current hold or delete operation to be applied. Specifying /MATCH=OR means that a message will match as long as at least one such criterion matches.

/MIN_LENGTH=n
By default, each substring to be searched for must be at least 24 bytes long (/MIN_LENGTH=24). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the /MIN_LENGTH qualifier to override this limit.

/THREADS=n
/NOTHREADS (default)
The /THREADS qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify /THREADS=*n*. The value *n* must be an integer in the range 1-8. The default is /NOTHREADS.

/VERBOSE
/NOVERBOSE (default)
The /VERBOSE qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

The following example shows holding all message files in the PMDF queue area that have the string "real estate" in the Subject: header and have the string "ownership.com" in the envelope From: address.

```
$ PMDF QCLEAN/MIN_LENGTH=11/SUBJECT="real estate"
/ENV_FROM="ownership.com"
%QM-I-QCLISTING, building a list of message files to scan from the
queue cache
%QM-I-SCANNING, scanning 72 message files
%QM-I-SCANNED, scanned 72 message files in 3.7500 seconds
(19.20 messages/second)
%QM-I-HELD, held 5 message files
```

QTOP—Display frequently occurring fields in PMDF queue area messages

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues.

SYNTAX

PMDF QTOP *[channel]*

Command Qualifiers	Defaults
/CONTENT[=offset-specifier]	*None*
/DATABASE	*/DATABASE*
/DIRECTORY_TREE	*/DATABASE*
/ENV_FROM[=offset-specifier]	*None*
/MIN_COUNT=n	*/MIN_COUNT=2*
/SUBJECT[=offset-specifier]	*/SUBJECT=(START=1,LENGTH=2147483647)*
/THREADS=n	*/NOTHREADS*
/TOP=n	*/TOP=20*
/VERBOSE	*/NOVERBOSE*

restrictions Privileges sufficient to read files in the PMDF channel queue directory tree, as well as read the PMDF queue cache database, are required.

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be scanned for string frequencies. * or ? wildcard characters may be used in the channel specification.

DESCRIPTION

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues. By default, only Subject: fields are shown (/SUBJECT). Use /ENV_FROM to display frequent envelope From: fields or /CONTENT to display frequent message contents. Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used.

The optional channel parameter restricts the scan to message files in the specified channel. The channel parameter may use * and ? wild cards.

By default, the top 20 most frequently occurring fields are shown (/TOP=20) provided that they occur 2 or more times (/MIN_COUNT=2). Use the /TOP and /MIN_COUNT qualifiers to alter this behavior. Also by default, only message files identified by the queue cache database are scanned (/DATABASE). Use the

/DIRECTORY_TREE qualifier to instead scan all message files actually present in the channel queue directory tree.

The /THREADS qualifier may be used to accelerate scanning on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous scanning threads, specify /THREADS=*n*. The value n must be in the range 1-8. The default is /NOTHREADS.

The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers accept the optional qualifiers START=n and LENGTH=n. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are /CONTENT=(START=1,LENGTH=256), /ENV_FROM=(START=1,LENGTH=2147483647), and /SUBJECT=(START=1,LENGTH=2147483647). Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

COMMAND QUALIFIERS

/CONTENT[=offset-specifier]
/ENV_FROM[=offset-specifier]
/SUBJECT[=offset-specifier]
The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (/SUBJECT). Use /ENV_FROM to display frequent envelope From: fields or /CONTENT to display frequent message contents. Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used.

The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers accept the optional qualifiers START=n and LENGTH=n. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are /CONTENT=(START=1,LENGTH=256), /ENV_FROM=(START=1,LENGTH=2147483647), and /SUBJECT=(START=1,LENGTH=2147483647). Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

/DATABASE (default)
/DIRECTORY_TREE
The /DATABASE qualifier, the default, specifies that only message files identified by the queue cache database be searched. Use the /DIRECTORY_TREE qualifier to instead search all message files actually present in the channel queue directory tree.

/MIN_COUNT=n
By default, a string must occur at least 2 times, /MIN_COUNT=2, in order to be displayed.

/THREADS=n
/NOTHREADS (default)
The /THREADS qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify /THREADS=*n*. The value *n* must be an integer in the range 1-8. The default is /NOTHREADS.

/TOP=n
By default, the top 20 most frequently occurring fields are shown, (/TOP=20).

/VERBOSE
/NOVERBOSE (default)
The /VERBOSE qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

The following example shows displaying the most frequently occurring Subject: and envelope From: addresses amongst messages in the PMDF queue area.

```
$ PMDF QTOP/SUBJECT/ENV_FROM
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Envelope From: addresses which occur 2 or more times
 Count  Envelope From: address
======  ========================================================================
    27
    10  owner-ex-list@example.com
     2  owner-test-list@example.com

Top 20 Subject: header lines which occur 2 or more times
 Count  Subject
======  ========================================================================
     6  Re: your ex-list posting
     2  Test posting to test-list
```

The following example shows displaying the most frequently occuring Subject: lines that occur 20 times or more, starting from 12 characters into the Subject: header value. This may be useful when trying to spot spam that inserts random characters at the beginning of the `Subject:` header value.

```
$ PMDF QTOP/SUBJECT=START=12/MIN_COUNT=15
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Subject: header lines which occur 15 or more times
 Count  Subject
======  ========================================================================
    25  ake money fast $$$
```

RESTART—Restart detached PMDF processes

Restart detached PMDF processes.

SYNTAX

PMDF RESTART *[component]*

Command Qualifiers	Defaults
/CLUSTER	*/NODE*
/NODE[=node]	*/NODE*
/ID=pid	*/NODE*

restrictions

SYSLCK privilege is required to restart detached PMDF processes.

prompts

Component: *component*

PARAMETERS

component
Optional parameter which specifies a specific PMDF component to be restarted, such as BN_SLAVE, CIRCUIT_CHECK, COUNTERS, DISPATCHER, FAX_RECEIVE, HTTP, IMAP (which restarts both the system mailbox server and the PMDF MessageStore mailbox server), IMAP_SERVER (the PMDF MessageStore server), POP_SERVER (the PMDF MessageStore server), POP3 (which restarts both the system mailbox server and PMDF MessageStore server), POPPASSD, or SMTP. Note that restarting the PMDF Service Dispatcher, *i.e.*, the DISPATCHER component, effectively restarts all the service components it handles, which may include HTTP, IMAP, IMAP_SERVER, POP_SERVER, POP3, POPPASSD, and SMTP servers. If no component name is given then all active components will be restarted.

DESCRIPTION

Detached PMDF processes should be restarted whenever the PMDF configuration is altered—these processes load information from the configuration once only and need to be restarted in order for configuration changes to become visible to them.

The RESTART utility is used to restart detached PMDF processes. The default is to restart processes on the node on which the command is executed. Use the /NODE qualifier with a specific node name to affect processes on a different node in the cluster; use the /CLUSTER qualifier to restart PMDF processes across the cluster; use the /ID qualifier to affect only a single process in the cluster. If no component parameter is specified, then all detached processes (including processes which use the PMDF API `PMDF_set_call_back` procedure) will be restarted. If

a component parameter is specified, then only detached processes associated with that component will be restarted. The standard component names are

Component	Description
BN_SLAVE	Detached processes which act as the Jnet Local Mail Delivery (LMD) daemon. Handles incoming local BITNET mail.
CIRCUIT_ CHECK	Detached process which monitors loopback message delivery.
COUNTERS	Detached processes which synchronize the channel counters.
DISPATCHER	PMDF multithreaded Service Dispatcher handling services such as SMTP, POP, IMAP, and HTTP servers.
FAX_RECEIVE	Detached processes which process incoming FAXes.
HTTP	HTTP server processes.
IMAP	This restarts both IMAP server processes serving out system mailboxes, and IMAP server processes serving out PMDF MessageStore mailboxes.
IMAP_SERVER	IMAP server processes serving out PMDF MessageStores mailboxes.
POP_SERVER	PMDF MessageStore POP3 server processes; that is, the processes serving out PMDF MessageStore and PMDF popstore mailboxes.
POP3	This restarts both POP3 server processes serving out system mailboxes, and POP3 server processes serving out PMDF MessageStore and PMDF popstore mailboxes.
POPPASSD	POPPASSD server processes.
SMTP	SMTP server processes.

Detached PMDF processes will restart as soon as is convenient. They restart by performing an orderly shutdown, exiting the image they are running, starting a new detached process running, and then exiting.

If there are no currently running process associated with a given component, then that component will not be restarted.

COMMAND QUALIFIERS

/CLUSTER
When the /CLUSTER qualifier is specified, the RESTART command will affect all nodes in the cluster.

/NODE[=node]
By default, the RESTART command affects only processes on the node on which the command is executed; this corresponds to specifying the /NODE qualifier without the optional node name value. To restart processes on a different node, specify the node name. The node name must be the SCS cluster node name.

/ID=pid
When the /ID qualifier is specified, only the process with the given process id (pid) is affected. Note that process id's are unique cluster wide and as such there is no need to also specify /NODE.

EXAMPLES

To restart cluster-wide all detached processes, simply use the command

```
$ PMDF RESTART/CLUSTER
```

To only restart, for instance, IMAP and POP3 servers on the current node, use the commands

```
$ PMDF RESTART IMAP
$ PMDF RESTART POP3
```

ERROR MESSAGES

SYSTEM-F-TIMEOUT, device timeout

After waiting two minutes for the component(s) in question to acknowledge the restart request, the PMDF RESTART utility returns control to the command line. The restart request is still outstanding and should be honored by the PMDF component(s) for which the command was issued; the utility has just given up waiting for the component(s) to immediately acknowledge the request as it (they) may be busy with existing activity.

RETURN—Return a mail message

Return (bounce) a mail message to its originator.

SYNTAX

PMDF RETURN *message-file-spec*

Command Qualifiers	Defaults
None.	*None.*

restrictions Postmaster privileges (write access to the PMDF channel queues) are required to use this utility.

PARAMETERS

message-file-spec
File specification of the message file to return. The specification may include wildcards.

DESCRIPTION

The RETURN utility returns a message to the message's originator. The returned message is in two parts. The first part explains the reason why the message is being returned; the text of the reason is contained in the file `return_bounced.txt` file located in the PMDF language-specific directory. The second part of the returned message contains the original message itself. Postmaster privileges (write access to the PMDF channel queues) are required to use this utility.

SHUTDOWN—Shut down detached PMDF processes

Shut down detached PMDF processes.

SYNTAX

PMDF SHUTDOWN *[component]*

Command Qualifiers	Defaults
/CLUSTER	*/NODE*
/NODE[=node]	*/NODE*
/ID=pid	*/NODE*

restrictions

SYSLCK privilege is required to shut down detached PMDF processes.

prompts

Component: *component*

PARAMETERS

component
Optional parameter which specifies a specific PMDF component to be shut down: DISPATCHER (which effectively shuts down *all* components handled by the PMDF Service Dispatcher), BN_SLAVE, CIRCUIT_CHECK, COUNTERS, FAX_RECEIVE, HTTP, IMAP (which shuts down both the system mailbox IMAP server and the PMDF MessageStore mailbox server), IMAP_SERVER (which shuts down the PMDF MessageStore mailbox server), POP_SERVER (which shuts down the PMDF MessageStore mailbox server), POP3 (which shuts down the system mailbox POP server and the PMDF MessageStore mailbox server), POPPASSD, SMTP, or the name of any Dispatcher service (as defined in the Dispatcher configuration file). If no component name is given then all active components will be shut down.

DESCRIPTION

The SHUTDOWN utility is used to shut down detached PMDF processes. The default is to shut down processes on the node on which the command is executed. Use the /NODE qualifier with a specific node name to shut down processes on a different node in the cluster; use the /CLUSTER qualifier to shut down detached PMDF processes cluster wide. For the FAX_RECEIVE process, the /ID qualifier may be used to affect only a single process in the cluster. If no component parameter is specified, then all detached processes (including processes which use the PMDF API `PMDF_set_call_back` procedure) will be shutdown. If a component parameter is specified, then only detached processes associated with that component will be shut down. The standard component names are

Component	Description
BN_SLAVE	Detached processes which act as the Jnet Local Mail Delivery (LMD) daemon. Handles incoming local BITNET mail.
CIRCUIT_ CHECK	Detached process which monitors loopback message delivery.
COUNTERS	Detached processes which synchronize the channel counters.
DISPATCHER	Multithreaded Service Dispatcher.
FAX_RECEIVE	Detached processes which process incoming FAXes.
HTTP	HTTP server processes.
IMAP	This shuts down both IMAP server processes serving out system mailboxes, and IMAP server processes serving out PMDF MessageStore mailboxes.
IMAP_SERVER	IMAP server processes serving out PMDF MessageStore mailboxes.
POP_SERVER	PMDF MessageStore POP3 server processes; that is, the processes serving out PMDF MessageStore and PMDF popstore mailboxes.
POP3	This shuts down both POP3 server processes serving out system mailboxes, and POP3 server processes serving out PMDF MessageStore and PMDF popstore mailboxes.
POPPASSD	POPPASSD server processes.
SMTP	SMTP server processes.

In addition, any Dispatcher service may be specified by name (that name used in the Dispatcher configuration file).

Detached PMDF processes will shutdown as soon as is convenient for the process to do so. They do so by performing an orderly shutdown, exiting the image they are running, and then exiting the process.

Note that in the case of BN_SLAVE, no Local Mail Delivery (LMD) daemon is left running, not even the default Jnet daemon.

COMMAND QUALIFIERS

/CLUSTER
When the /CLUSTER qualifier is specified, the SHUTDOWN command will affect all nodes in the cluster.

/NODE[=node]
By default, the SHUTDOWN command affects only processes on the node on which the command is executed; this corresponds to specifying the /NODE qualifier without the optional node name value. To shut down processes on a different node, specify the node name. The node name must be the SCS cluster node name.

/ID=pid
For a FAX_RECEIVE process, the /ID qualifier may be used to specify that only the FAX_RECEIVE process with the given process id (pid) is to be affected. The /ID qualifier is only valid for FAX_RECEIVE; it is not valid for other processes such as Dispatcher services. Note that process id's are unique cluster wide and as such there is no need to also specify /NODE.

EXAMPLES

To shutdown cluster-wide all detached processes, simply use the command

```
$ PMDF SHUTDOWN/CLUSTER
```

To only shutdown, for instance, IMAP and POP3 servers on the current node, use the commands

```
$ PMDF SHUTDOWN IMAP
$ PMDF SHUTDOWN POP3
```

ERROR MESSAGES

```
SYSTEM-F-TIMEOUT, device timeout
```

After waiting two minutes for the component(s) in question to acknowledge the shutdown request, the PMDF SHUTDOWN utility returns control to the command line. The shutdown request is still outstanding and should be honored by the PMDF component(s) for which the command was issued; the utility has just given up waiting for the component(s) to immediately acknowledge the request as it (they) may be busy with existing activity.

STARTUP—Start up detached PMDF processes

Start up detached PMDF processes.

SYNTAX

PMDF STARTUP *component*

Command Qualifiers	Defaults
None.	*None.*

restrictions

SYSLCK privilege is required to start up detached PMDF processes.

prompts

Component: *component*

PARAMETERS

component
Required parameter which specifies a specific PMDF component to be started: DISPATCHER (which effectively starts *all* components handled by the PMDF Service Dispatcher), CIRCUIT_CHECK, COUNTERS, FAX_RECEIVE.

DESCRIPTION

The STARTUP utility is used to start up detached PMDF processes such as the PMDF Service Dispatcher, or specific service processes such as FAX_RECEIVE. The standard component names are

Component	Description
CIRCUIT_ CHECK	Detached process which monitors loopback message timings.
COUNTERS	Detached processes which synchronize the channel counters.
DISPATCHER	Multithreaded Service Dispatcher.
FAX_RECEIVE	Detached processes which process incoming FAXes.

Note the services handled by the PMDF multithreaded Service Dispatcher must be started by starting the PMDF Service Dispatcher; only services *not* being handled by the PMDF Service Dispatcher can be individually started via the PMDF STARTUP utility. The Service Dispatcher may be configured to handle various services, *e.g.*, and the multithreaded HTTP, IMAP (the system mailbox IMAP server), IMAP_SERVER (the PMDF MessageStore mailbox server), POP_ SERVER (the PMDF MessageStore mailbox server), POP3 (the system mailbox POP server), POPPASSD, and SMTP servers. See Chapter 11 for details.

EXAMPLES

The following command starts the PMDF Service Dispatcher:

```
$ PMDF STARTUP DISPATCHER
```

TEST/MAPPING—Test a mapping table

Test a mapping table in the mapping file.

SYNTAX

PMDF TEST/MAPPING *[input-string]*

Command Qualifiers	Defaults
/FLAGS=(a,b,c,...)	*/NOFLAGS*
/IMAGE_FILE=file-spec	*/IMAGE_FILE=PMDF_CONFIG_DATA*
/MAPPING_FILE=file-spec	*/MAPPING_FILE=PMDF_MAPPING_FILE*
/OPTION_FILE=file-spec	*/OPTION_FILE=PMDF_OPTION_FILE*
/TABLE=table-name	*None*

restrictions *None.*

prompts Enter table name: *table-name*

PARAMETERS

input-string
Optional input string to map.

DESCRIPTION

TEST/MAPPING may be used to test the behavior of a mapping table in the mapping file. The result of mapping an input string will be output along with information about any metacharacters specified in the output string.

If an input string is supplied on the command line, then only the result of mapping that input string will be output. If no input string is specified TEST/MAPPING will enter a loop, prompting for an input string, mapping that string, and prompting again for another input string. TEST/MAPPING will exit when a CTRL/Z is entered.

COMMAND QUALIFIERS

/FLAGS=(a,b,c,...)
/NOFLAGS
The /FLAGS qualifier is used to specify particular flags to set during the mapping testing; for instance, the E (envelope), B (header/body), or I (message id) flags when testing a REVERSE mapping.

/IMAGE_FILE[=filename]
/NOIMAGE_FILE
The /IMAGE_FILE qualifier serves two purposes. The first is when /NOIMAGE_FILE is specified; this instructs TEST/MAPPING to ignore any compiled mapping information unconditionally and to read mapping information from the mapping file itself.

When the /IMAGE_FILE qualifier is specified without an optional file name, PMDF will load the compiled configuration file PMDF_CONFIG_DATA. If, instead, a file name is specified then that file, which is expected to be a compiled configuration image, will be loaded instead.

/MAPPING_FILE=filename
This qualifier instructs TEST/MAPPING to use the specified mapping file rather than the default mapping file, PMDF_MAPPING_FILE.

This qualifier has no effect unless /NOIMAGE_FILE is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of mapping file.

/OPTION_FILE=filename
/NOOPTION_FILE
This qualifier instructs TEST/MAPPING to use the specified option file rather than the default option file PMDF_OPTION_FILE.

This qualifier has no effect unless /NOIMAGE_FILE is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of option file.

Use of the qualifier /NOOPTION_FILE will prevent the file PMDF_OPTION_FILE from being read in when there is no compiled configuration.

/TABLE=table-name
This qualifier specifies the name of the mapping table to test. If this qualifier is not specified, then TEST/MAPPING will prompt for the name of a table to use.

EXAMPLES

In the following example, the sample PAGER mapping is tested. The /MAPPING_FILE qualifier is used to select the mapping file `pager_table.sample` instead of the default mapping file.

```
$ PMDF TEST/MAPPING/NOIMAGE/MAPPING_FILE=PMDF_TABLE:pager_table.sample
Enter table name: PAGER
Input string: H|From: "Daniel C. Newman"
<dan@example.com> (Doof City)
Output string: H|F:dan
Output flags: [0, 1, 2, 'Y' 89]
Input string: ^Z
$
```

TEST/MATCH—Test a mapping wildcard pattern

Test a mapping wildcard pattern.

SYNTAX

PMDF TEST/MATCH

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

prompts

Pattern:	*mapping-pattern*
Target:	*target-string*

PARAMETERS *None.*

DESCRIPTION

TEST/MATCH may be used to test wildcard and glob matching, such as in a mapping pattern.

When invoked, TEST/MATCH prompts for a pattern and then for a target string to compare against the pattern, and will output whether or not the target string matched and if it did match, which characters in the target string matched which wildcard or glob of the pattern. TEST/MATCH will loop, prompting for input, until exitted with a CTRL/Z.

EXAMPLES

In the following example, the sample mapping pattern `$[ax1]*@*.example.com` is tested for several sample target strings.

```
$ PMDF TEST/MATCH
Pattern: $[ax1]*@*.example.com
  [  1S] cglob [1ax]
  [  2] "@"
  [  3S] glob, req 109, reps 2
  [  4] "."
  [  5] "a"
  [  6] "c"
  [  7] "m"
  [  8] "e"
  [  9] "."
  [ 10] "c"
  [ 11] "o"
  [ 12] "m"
Target: xx11a@sys1.example.com
Match.
0 - xx11a
1 - sys1
Pattern: $[ax1]*@*.example.com
  [  1S] cglob [1ax]
  [  2] "@"
  [  3S] glob, req 109, reps 2
  [  4] "."
  [  5] "a"
  [  6] "c"
  [  7] "m"
  [  8] "e"
  [  9] "."
  [ 10] "c"
  [ 11] "o"
  [ 12] "m"
Target: 12a@node.example.com
No match.
Pattern: $[ax1]*@*.example.com
  [  1S] cglob [1ax]
  [  2] "@"
  [  3S] glob, req 109, reps 2
  [  4] "."
  [  5] "a"
  [  6] "c"
  [  7] "m"
  [  8] "e"
  [  9] "."
  [ 10] "c"
  [ 11] "o"
  [ 12] "m"
Target: 1xa@node.example.com
Match.
0 - 1xa
1 - node
Pattern: ^Z
$
```

TEST/REWRITE—Test address rewriting

Test address rewriting specified by a PMDF configuration

SYNTAX

PMDF TEST/REWRITE *[test-address[,...]]*

Command Qualifiers	Defaults
/ALIAS_FILE=file-spec	*/ALIAS_FILE=PMDF_ALIAS_FILE*
/CHANNEL	*/CHANNEL*
/CHECK_EXPANSIONS	*/NOCHECK_EXPANSIONS*
/CONFIGURATION_FILE=file-spec	*/CONFIGURATION_FILE=PMDF_CONFIG_FILE*
/DATABASE=database-list	*See text*
/DEBUG	*/NODEBUG*
/DELIVERY_RECEIPT	*See text*
/DESTINATION_CHANNEL=channel	*None*
/FILTER	*/NOFILTER*
/FROM=address	*/FROM=postmaster@localhost*
/GREY=setting	*/GREY=0*
/IMAGE_FILE=file-spec	*/IMAGE_FILE=PMDF_CONFIG_DATA*
/LOCAL_ALIAS=value	*None*
/MAPPING_FILE=file-spec	*/MAPPING_FILE=PMDF_MAPPING_FILE*
/OPTION_FILE=file-spec	*/OPTION_FILE=PMDF_OPTION_FILE*
/READ_RECEIPT	*See text*
/REPROCESSING	*/REPROCESSING*
/RESTRICTED=setting	*/RESTRICTED=0*
/SOURCE_CHANNEL=channel	*/SOURCE_CHANNEL=L*

restrictions *None.*

prompts Address: *test-address[,...]*

PARAMETERS

test-address
Optional parameter specifying one or more addresses to rewrite.

DESCRIPTION

TEST/REWRITE provides a straightforward test facility for examining PMDF's address rewriting and channel mapping process without actually sending any message. Various qualifiers can be used to control whether TEST/REWRITE uses the configuration text files or the compiled configuration (if present), the amount of output produced, and so on.

If one or more test addresses are specified on the command line, TEST/REWRITE applies PMDF address rewriting to those addresses, reports the results, and exits. If no test address is specified TEST/REWRITE will enter a loop, prompting for addresses, rewriting them, and prompting again for more addresses. TEST/REWRITE will exit when a CTRL/Z is entered.

When testing rewriting of a alias corresponding to a mailing list which has an AUTH_ or CANT_ type of named parameter controlling who is authorized to post to the list, or when testing rewriting when SEND_ACCESS or related mapping tables are in effect, note that by default TEST/REWRITE uses as the posting address the return address of the local postmaster as specified by the RETURN_ ADDRESS option in the PMDF option file. To specify a different posting address for the rewriting process, use the /FROM qualifier.

COMMAND QUALIFIERS

/ALIAS_FILE=filename
TEST/REWRITE normally consults the default alias file PMDF_ALIAS_FILE during the rewriting process. The /ALIAS_FILE qualifier specified an alternate file for TEST/REWRITE to use.

This qualifier has no effect unless /NOIMAGE_FILE is specified or no compiled configuration exists; use of any compiled configuration precludes reading any sort of alias file.

/CHANNEL (default)
/NOCHANNEL
This qualifier controls whether the utility outputs detailed information, *e.g.*, channel flags, regarding the channel an address matches.

/CHECK_EXPANSIONS
/NOCHECK_EXPANSIONS (default)
This qualifier controls checking of alias address expansion. Normally PMDF considers the expansion of an alias to have been successful if any of the addresses the alias expands to are legal. The /CHECK_EXPANSIONS qualifier causes a much stricter policy to be applied; TEST/REWRITE checks each expanded address in detail and reports a list of any addresses, expanded or otherwise, that fail to rewrite properly. For addresses that match the L channel, PMDF also performs validity checks.

/CONFIGURATION_FILE=filename
TEST/REWRITE normally consults the default configuration file PMDF_CONFIG_ FILE during the rewriting process. The /CONFIGURATION_FILE qualifier specifies an alternate file to use in place of the file PMDF_CONFIG_FILE.

This qualifier has no effect unless /NOIMAGE_FILE is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of configuration file.

/DEBUG
/NODEBUG (default)
The address rewriting process is capable of producing additional, detailed explanations of what actions are taken and why. The /DEBUG qualifier enables this output; it is disabled by default.

/DATABASE=database-list
TEST/REWRITE normally consults the usual PMDF databases during its operation. This qualifier is used to either disable references to various databases or to redirect the database paths to nonstandard locations.

The allowed list items are ALIAS, NOALIAS, PERSONAL_ALIAS, NOPERSONAL_ALIAS, DOMAIN, NODOMAIN, FORWARD, NOFORWARD, GENERAL, NOGENERAL, REVERSE, and NOREVERSE. The list items beginning with "NO" disable use of the corresponding database. The remaining items require an associated value, which is taken to be the name of that database.

/DELIVERY_RECEIPT
/NODELIVERY_RECEIPT
The /DELIVERY_RECEIPT and /NODELIVERY_RECEIPT qualifiers, which explicitly set the corresponding receipt request flags, can be useful when testing the handling of receipt requests when rewriting forwarded addresses or mailing lists.

/DESTINATION_CHANNEL=channel
The /DESTINATION_CHANNEL qualifier controls what destination or target channel TEST/REWRITE rewrites addresses for. Some address rewriting is destination channel specific; this qualifier allows control of the assumed destination channel.

/FILTER
/NOFILTER (default)
The /FILTER qualifier may be used to have PMDF TEST/REWRITE output any filters (personal mailbox, channel, or system) applying for the address in question.

/FROM=address
/NOFROM
This qualifier controls what envelope From: address is used for access control probes and mailing list access probes. If the /FROM qualifier is omitted, then the address used for access checks is the postmaster return address. Specifying either /FROM=<> or /NOFROM tells the utility to use an empty envelope From: address for access checks.

/GREY=setting
/NOGREY (default)
This qualifier controls the setting of the Grey Book flag. By default, this flag has value 0. When set to 1, /GREY=1, the Grey Book flag will be set on and addresses will be rewritten using the Grey Book format.

This flag is used to force rewriting of address in accordance with the JANET (Grey Book) specifications. The most significant effect is that domain specifications appear in reverse order, *e.g.,* edu.claremont.ymir and not ymir.claremont.edu. See Section 2.3.4.83 for further details.

Grey Book address formats are not currently used in PMDF, so this qualifier's usefulness is problematic at best.

/IMAGE_FILE[=filename]
/NOIMAGE_FILE
The /IMAGE_FILE qualifier serves two purposes. The first is when /NOIMAGE_FILE is specified; this instructs TEST/REWRITE to ignore any compiled configuration unconditionally and to read configuration information from the various text files instead.

When the /IMAGE_FILE qualifier is specified without an optional file name, PMDF TEST/REWRITE will load the compiled configuration from the file PMDF_CONFIG_DATA. If, instead, a file name is specified then TEST/REWRITE will load the compiled configuration from the specified file.

/LOCAL_ALIAS=value
/NOLOCAL_ALIAS (default)
This qualifier controls the setting of an alias for the local host. PMDF supports multiple "identities" for the local host; the local host may have a different identity on each channel. This qualifier may be used to set the local host alias to the specified value; appearances of the local host in rewritten addresses will be replaced by this value.

/MAPPING_FILE[=filename]
/NOMAPPING_FILE
This qualifier instructs TEST/REWRITE to use the specified mapping file rather than the default mapping file named by the PMDF_MAPPING_FILE logical name, usually `PMDF_TABLE:mappings.`

This qualifier has no effect unless /NOIMAGE_FILE was specified or no compiled configuration exists; use of any compiled configuration will preclude reading the mapping file.

Use of the /NOMAPPING_FILE qualifier will prevent the PMDF_MAPPING_FILE file from being read in when there is no compiled configuration.

/OPTION_FILE=filename
/NOOPTION_FILE
This qualifier instructs TEST/REWRITE to use the specified option file rather than the default option file PMDF_OPTION_FILE.

This qualifier has no effect unless /NOIMAGE_FILE is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of option file.

Use of the qualifier /NOOPTION_FILE will prevent the file PMDF_OPTION_FILE from being read in when there is no compiled configuration.

/READ_RECEIPT
/NOREAD_RECEIPT
The /READ_RECEIPT and /NOREAD_RECEIPT qualifiers, which explicitly set the corresponding receipt request flags, can be useful when testing the handling of receipt requests when rewriting forwarded addresses or mailing lists.

/REPROCESSING (default)
/NOREPROCESSING
This qualifier allows the utility to display the contents of a mailing list which uses the `[REPROCESS]` named parameter in its alias definition.

/RESTRICTED=value
/NORESTRICTED
This qualifier controls the setting of the restricted flag. By default, this flag has value 0. When set to 1, /RESTRICTED=1, the restricted flag will be set on and addresses will be rewritten using the restricted mailbox encoding format recommend by RFC1137.

This flag is used to force rewriting of address mailbox names in accordance with the RFC1137 specifications; see Section 2.3.4.57 for further details.

/SOURCE_CHANNEL=channel
The /SOURCE_CHANNEL qualifier controls what source channel to rewrite addresses for. Some address rewriting is source channel specific; TEST/REWRITE normally pretends that the channel source it is rewriting for is the local channel, L.

EXAMPLES

This example shows the typical output generated by TEST/REWRITE. Perhaps the single most important piece of information generated by TEST/REWRITE is the last few lines of the TEST/REWRITE output, ❻, which give the channel to which TEST/REWRITE would submit a message with the specified test address and the form in which the test address would be rewritten for that channel. This output is invaluable when debugging configuration problems.

```
$ PMDF TEST/REWRITE DAN@EXAMPLE.COM
❶  forward channel        = tcp_local
   channel description    =
   channel user filter    =
   dest channel filter    =
   source channel filter  =
❷  channel flags #0       = BIDIRECTIONAL SINGLE_SYSTEM IMMNORMAL NOSERVICEALL
   channel flags #1       = SMTP RANDOMMX MAYTLS DEFAULT
   channel flags #2       = NOLOCALPOST POSTHEADBODY HEADERINC NOEXPROUTE
   channel flags #3       = LOGGING NOGREY NORESTRICTED
   channel flags #4       = EIGHTNEGOTIATE NOHEADERTRIM NOHEADERREAD RULES
   channel flags #5       = MASTER_DEBUG
   channel flags #6       = LOCALUSER REPORTHEADER
   channel flags #7       = SWITCHCHANNEL NOREMOTEHOST DATEFOUR DAYOFWEEK
   channel flags #8       = NODEFRAGMENT EXQUOTA REVERSE NOCONVERT_OCTET_STREAM
   channel flags #9       = NOTHURMAN INTERPRETENCODING INCLUDEFINAL RECEIVEDFROM
   linelength             = 998
   addrsperfile           = 127
   channel env addr type  = SOURCEROUTE
   channel hdr addr type  = SOURCEROUTE
```

```
❸   channel official host  = TCP-DAEMON
    channel local alias    =
❹   channel queue name     = MAIL_TCP_BATCH
    channel after param    = 60
    channel daemon name    = fw.example.com
    channel user name      =
    urgentnotices          = 1 4 8 12
    normalnotices          = 1 4 8 12
    nonurgentnotices       = 1 4 8 12
❺   channel rightslist ids =
❻   backward channel       = tcp_local
    header To: address     = DAN@EXAMPLE.COM
    header From: address   = DAN@EXAMPLE.COM
    envelope To: address   = DAN@EXAMPLE.COM  (route (TCP-DAEMON,TCP-DAEMON))
    envelope From: address = DAN@EXAMPLE.COM
    name                   =
    mbox                   = DAN
  Extracted address action list:
      DAN@EXAMPLE.COM
  Extracted 733 address action list:
      DAN@EXAMPLE.COM
  Address list expansion:
  0 expansion total.
  Expanded address:
     DAN@EXAMPLE.COM
❼ Submitted address list:
    tcp_local
      DAN@EXAMPLE.COM (EXAMPLE.COM) *NOTIFY FAILURES* *NOTIFY DELAYS*

❽ Submitted notifications list:
```

❶ The channel to which, after rewriting as an envelope To: address, the address is mapped.

❷ The flags set for the channel indicated in ❶. These flags are controlled by the channel keywords on the first line of the channel control block for the specified channel. Any unknown keywords—keywords which may have been mistyped—will be interpreted as rightslist identifiers and will appear on the line ❹.

❸ The channel's official host name as specified on the second line of the channel control block for the channel indicated in ❶.

❹ The channel queue name, channel after parameter, and channel daemon name correspond to values specified via the `queue`, `after`, and `daemon` channel keywords, respectively. If no such keyword is present on the channel, then no such line of output will be displayed.

❺ Any items appearing on the first line of the channel block which were not channel keywords are interpreted as rightslist identifiers. Any rightslist identifiers so specified for the channel are listed on this line.

❻ The channel which the address would match as an envelope From: address.

❼ The channel to which a message with the address DAN@EXAMPLE.COM would be queued and the envelope To: address which would be used. Here, the message would be submitted to the TCP/IP channel, tcp_local, using the address DAN@EXAMPLE.COM. Other information appearing here might include an explicit Errors-to: address, which, if present, appears enclosed in square brackets; or notations such as `*RR*` or `*NRR*`, indicating whether or not a message is flagged

for read receipts, or notations such as *NOTIFY FAILURES*, *NOTIFY DELAYS*, *NOTIFY SUCCESSES*, *etc.*, indicating the message's delivery receipt mechanism and flagging.

❽ Notification addresses (reserved for future use).

ERROR MESSAGES

Usually errors reported by PMDF TEST/REWRITE are not actually errors regarding PMDF TEST/REWRITE in particular, but rather are the utility warning of an underlying configuration problem. For instance, "Error in mm_init: ..." sorts of errors; see for a discussion of many such general error messages.

`Address list error -- unknown host or domain:`

The domain name in the specified address did not rewrite to any PMDF channel. Check that the domain name was correctly spelled. If the domain name was correct, then most likely you need a new (or changed) rewrite rule in the PMDF configuration file to handle that domain name; see

`Unknown rightslist identifier ... found on channel ...`

You do not have the specified rightslist identifier. Check that it is truly intended to be present as a rightslist identifier, rather than simply being a misspelled channel keyword.

TEST/URL—Test an LDAP query URL

Test an LDAP query URL.

SYNTAX

PMDF TEST/URL *[ldap-url]*

Command Qualifiers	Defaults
/DEBUG	*/NODEBUG*

restrictions *None.*

prompts _URL: *ldap-url*

PARAMETERS

ldap-url
LDAP URL to try resolving.

DESCRIPTION

Test an LDAP query URL.

Note that the LDAP server to query is controlled by the setting of the PMDF options LDAP_HOST and LDAP_PORT in the PMDF option file; see Section 7.3.2.

COMMAND QUALIFIERS

/DEBUG
/NODEBUG (default)
The testing process is capable of producing additional debug output. The /DEBUG qualifier enables this output; it is disabled by default.

EXAMPLES

This example shows a sample query for a site example.com that has set the LDAP_HOST and LDAP_PORT options in the PMDF option file to point to an LDAP server containing e-mail addresses in a `mail` attribute.

```
$ pmdf test/url "ldap:///dc=example,dc=com?mail?sub?sn=doe"
URL> Jane.Doe@example.com
URL> John.Doe@example.com
```

VERSION—Print PMDF version number

Print PMDF version number.

SYNTAX

PMDF VERSION

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

PARAMETERS

None.

DESCRIPTION

VERSION prints out the PMDF installed version number, and displays the system's architecture type and operating system version number.

EXAMPLES

To check what version of PMDF you are running, issue the command:

```
$ PMDF VERSION
%PMDF-I-VERSION, PMDF version is PMDF V6.6
        AlphaServer 4X00 5/466 4MB running OpenVMS Alpha V8.3
        PMDF_SHARE_LIBRARY version V6.6; linked 13:06:23, Feb  4 2012
```

29.2 Interactive Utilities on OpenVMS

PMDF has eight interactive utilities on OpenVMS, CONFIGURE, DB, MAIL, MOVEIN, MSGSTORE, POPSTORE, and QM. Further details on use of CONFIGURE may be found in the *PMDF Installation Guide, OpenVMS Edition*. DB and MAIL are both described in the *PMDF User's Guide, OpenVMS Edition*. POPSTORE and MOVEIN are described in the *PMDF popstore & MessageStore Manager's Guide*. MSGSTORE is described in the *PMDF popstore & MessageStore Manager's Guide*. QM is described in Section 29.2.1.

29.2.1 QM: Queue Management Utility

PMDF QM is a utility program which allows inspection and manipulation of queued messages. PMDF QM has two modes: maintenance mode and user mode. Maintenance mode can be used to inspect and manipulate the channel queue directories and the messages contained in them. Privileges sufficient to read, create, and delete files in the channel queue directory tree as well as read and update the queue cache database are required to use maintenance mode. User mode is a very restricted version of maintenance mode which allows unprivileged users to read their own messages from the queues and to return them (bounce them) back to their originator if desired. Users' own messages are messages which they themselves have sent or were posted to a list they own. They are not messages destined for the user. User mode is documented in the *PMDF User's Guide, OpenVMS Edition*.

To run PMDF QM in maintenance mode, issue the command

```
$ PMDF QM/MAINTENANCE
```

Use the EXIT or QUIT command to exit PMDF QM.

The commands accepted by this utility[2] are summarized in Table 29–2 below.

[2] Note that the commands accepted by the PMDF QM utility are a subset of those used by the old `qm.com` command procedure provided on OpenVMS. However, all of the functionality of that old procedure has been retained with redundant commands removed (*e.g.*, TIME and READHELD).

Table 29–2 Summary of PMDF QM Maintenance Mode Commands

CLEAN	Hold or delete message files matching specified criteria
COUNTERS	Control aspects of the channel counter caches and database
DATE	Show current date and time
DELETE	Irrevocably delete the specified messages
DIRECTORY	List currently queued messages
EDIT_FAX	Edit a queued PMDF-FAX message
HELD	List messages which have been marked as held
EXIT	Exit the utility
QUIT	Exit the utility
HELP	Obtain help
HISTORY	Display message delivery history information
HOLD	Mark a message as held
READ	Display message envelope and header information
RELEASE	Release held message
RETURN	Return a message to its originator
SPAWN	Spawn a subprocess
SUMMARIZE	Display a summary listing of message files
TOP	Display frequently occurring strings from PMDF queue area message files
VIEW	Control whether the channel queue directory tree or queue cache database is viewed

CLEAN

Hold or delete message files from the PMDF queue area that contain specified substrings in their envelope From: address, Subject: header, or message content.

SYNTAX

CLEAN *[channel]*

Command Qualifiers	Defaults
/CONTENT=substring	*None*
/DATABASE	*See text*
/DELETE	*/HOLD*
/DIRECTORY_TREE	*See text*
/ENV_FROM=substring	*None*
/HOLD	*/HOLD*
/MATCH=keyword	*/MATCH=AND*
/MIN_LENGTH=n	*/MIN_LENGTH=24*
/SUBJECT=substring	*None*
/THREADS=n	*/NOTHREADS*
/VERBOSE	*/NOVERBOSE*

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be searched for matching messages. * or ? wildcard characters may be used in the channel specification.

DESCRIPTION

Hold or delete message files containing specific substrings in their envelope From: address, Subject: line, or content. By default, message files are held (/HOLD). Specify /DELETE to instead delete matching message files. The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers are used to specify the substrings for which to search.

Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used. The /MATCH qualifier controls whether a message file must contain all (/MATCH=AND, the default) or only one of (/MATCH=OR) the specified substrings in order to be held or deleted. The default is /MATCH=AND.

By default, each substring to be searched for must be at least 24 bytes long (/MIN_LENGTH=24). This is a safety measure: the longer the substring, the less likely the chance of false “hits”. Use the /MIN_LENGTH qualifier to override this limit. The message files searched may be either all those present in the channel queue directory tree, or only those files with entries in the queue

cache database. Use either the VIEW command or the /DIRECTORY_TREE or /DATABASE qualifier to control which files are searched.

The optional channel parameter restricts the search to message files in the specified channel. The channel parameter may use * and ? wild cards.

The /THREADS qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searchingg threads, specify /THREADS=*n*. The value *n* must be in the range 1-8. The default is /NOTHREADS.

COMMAND QUALIFIERS

/CONTENT=substring
/ENV_FROM=substring
/SUBJECT=substring
The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers are used to specify the substrings for which to search. Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used. When a combination of such qualifiers is used, the /MATCH qualifier controls whether the qualifiers are interpreted as further restrictions (/MATCH=AND), or as alternatives (/MATCH=OR).

/DATABASE
/DIRECTORY_TREE
Controls whether the message files searched are only those with entries in the queue cache database, /DATABASE, or all message files actually present in the channel queue directory tree, /DIRECTORY_TREE.

When neither /DATABASE nor /DIRECTORY_TREE is specified, then the "view" selected with the VIEW command will be used. If no VIEW command has been issued, then /DIRECTORY_TREE is assumed.

/DELETE
/HOLD (default)
/HOLD is the default and means that matching message files will be held. Specify /DELETE to instead delete matching message files.

/MATCH=keyword
The default is /MATCH=AND, meaning that any criteria specified by /CONTENT, /ENV_FROM, and /SUBJECT qualifiers must all match in order for the current hold or delete operation to be applied. Specifying /MATCH=OR means that a message will match as long as at least one such criterion matches.

/MIN_LENGTH=n
By default, each substring to be searched for must be at least 24 bytes long (/MIN_LENGTH=24). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the /MIN_LENGTH qualifier to override this limit.

/THREADS=n
/NOTHREADS (default)
The /THREADS qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads.

To run *n* simultaneous searching threads, specify /THREADS=*n*. The value *n* must be an integer in the range 1-8. The default is /NOTHREADS.

/VERBOSE
/NOVERBOSE (default)
The /VERBOSE qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

The following example shows holding all message files in the PMDF queue area that have the string "real estate" in the Subject: header and have the string "ownership.com" in the envelope From: address.

```
qm.maint> CLEAN/MIN_LENGTH=11/SUBJECT="real estate"
/ENV_FROM="ownership.com"
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 72 message files
%QM-I-SCANNED, scanned 72 message files in 3.7500 seconds (19.20 messages/second)
%QM-I-HELD, held 5 message files
```

COUNTERS CLEAR

Clear the node-specific, in-memory cache of counters.

SYNTAX

COUNTERS CLEAR

Command Qualifiers	Defaults
/ASSOCIATIONS	*/ASSOCIATIONS*
/CHANNELS	*/CHANNELS*

PARAMETERS

None.

DESCRIPTION

To clear (zero) the counters in a node-specific, in-memory cache, issue the COUNTERS CLEAR command on that particular node. The command creates the node-specific, in-memory section of association and channel counters if it does not already exist. Then it zeros all fields in the in-memory section. Note that the counters will be zeroed without first merging their values into the cluster-wide database of channel counters. If a cluster-wide, on-disk database does not already exist, a new one will be created. Finally, the fields in the on-disk database for numbers of stored messages, message recipients, and message volumes are set based on the entries in the PMDF queue cache database.

Either the association counters, or channel counters, or both, may be cleared. The default is to clear both association and channel counters.

If you want to update the on-disk database with the old in-memory values before clearing them, then you should issue a COUNTERS SYNCHRONIZE command before issuing the COUNTERS CLEAR command.

COMMAND QUALIFIERS

/ASSOCIATIONS (default)
/NOASSOCIATIONS
This qualifier specifies whether to clear the in-memory cache of association counters.

/CHANNELS (default)
/NOCHANNELS
This qualifier specifies whether to clear the in-memory cache of channel counters.

COUNTERS CRDB

Create a cluster-wide database of accumulated association and channel counters.

SYNTAX

COUNTERS CRDB

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

None.

DESCRIPTION

A new, cluster-wide database of channel counters can be created with the COUNTERS CRDB command. The new database will have all counters zeroed except for the count of messages stored in each channel. Those counts will be determined by entries in the PMDF queue cache database. In addition, if an in-memory section for association and channel counters on this node does not already exist, it will be created as well.

Once the on-disk, cluster-wide database exists, you may use the COUNTERS SYNCHRONIZE command to merge the information from the node-specific, in-memory cache of counters into the on-disk database.

COUNTERS SHOW

Display the contents of the cluster-wide database of channel counters.

SYNTAX

COUNTERS SHOW *[channel]*

Command Qualifiers	Defaults
/HEADER	*/HEADER*
/OUTPUT=file-spec	*None*
/SYNCHRONIZE	*/SYNCHRONIZE*
/TIMEOUT=seconds	*/TIMEOUT=120*

PARAMETERS

channel
Optional channel name indicating the channel(s) for which to show counters. May contain wildcards.

DESCRIPTION

The contents of the cluster-wide channel counter database may be displayed with the COUNTERS SHOW command. By default, before the counters are displayed, an implicit COUNTERS SYNCHRONIZE command will be executed, to attempt to synchronize each node-specific cache with the main cluster-wide database. Specify /NOSYNCHRONIZE to merely display the current contents of the database without first synchronizing the node-specific caches.

Note that SYSLCK privilege is required to perform the synchronization step.

Note that the output of PMDF QM's COUNTERS SHOW command is currently not as detailed as the output of the DCL level PMDF COUNTERS/SHOW command.

COMMAND QUALIFIERS

/HEADER (default)
/NOHEADER
Controls whether or not a header line describing each column in the table of counters is output.

/OUTPUT=file-spec
Direct the output to the specified file. By default the output appears on your display.

/SYNCHRONIZE (default)
/NOSYNCHRONIZE
Before displaying the counters, attempt to synchronize each of the node-specific caches with the cluster-wide database. Specify /NOSYNCHRONIZE to skip this synchronization step.

/TIMEOUT=seconds
By default, QM will wait upwards of 120 seconds for the node-specific caches to be synchronized with the cluster-wide database. Should the synchronization step not be completed before the specified time period, then QM will stop waiting and proceed to display the information from the database. You may specify a different period of time to wait with the /TIMEOUT qualifier.

This qualifier has no effect when /NOSYNCHRONIZE is specified.

EXAMPLES

To display the counters information for all TCP/IP channels, use the command

```
qm.maint> COUNTERS SHOW *tcp_*
Channel                      Messages  Recipients      Blocks
-------------------------  ----------  ----------  ----------
tcp_local
    Received                       33          41          95
    Stored                          0           0           0
    Delivered                      33          41          95
    Submitted                       1           1           3
tcp_internal
    Received                      632         758        1453
    Stored                          1           2          10
    Delivered                     631         756        1443
    Submitted                       3           6          12
qm.maint>
```

COUNTERS SYNCHRONIZE

Synchronize each of the node-specific, in-memory caches of channel counters with the cluster-wide database.

SYNTAX

COUNTERS SYNCHRONIZE

Command Qualifiers	Defaults
/TIMEOUT=seconds	*/TIMEOUT=120*

PARAMETERS

None.

DESCRIPTION

To synchronize each of the node-specific, in-memory cache of channel counters with the cluster-wide database, issue a COUNTERS SYNCHRONIZE command. The command will not return control back to you until either all the caches have been synchronized or a "timeout" period has elapsed. Should the timeout period elapse, then control will be returned to you. However, the synchronization process will continue in the background. Use the /TIMEOUT qualifier to adjust the timeout period which has a default value of 120 seconds.

Note that SYSLCK privilege is required to use this command.

Note that the COUNTERS SYNCHRONIZE command signals each PMDF counters synchronization process in the cluster to perform the synchronization—there should be one such process on each node running PMDF. Note that on each node, the synchronization can only be performed if the PMDF counters synchronization process is running on that node.

Assuming that the PMDF counters synchronization process is running on each node, then for each node the node-specific, in-memory cache will be created, if it does not already exist. If the cluster-wide, on-disk database does not exist, it will be created. The in-memory cache values will be used to update the on-disk database, and then the on-disk database values for stored messages, recipients, and volume will be set by scanning the PMDF queue cache database.

COMMAND QUALIFIERS

/TIMEOUT=seconds

By default, QM will wait upwards of 120 seconds for the node-specific caches to be synchronized. Should the synchronizations not be completed before the specified time period, QM will return control to you prompting you for another command. The synchronization process will, however, continue in the background.

COUNTERS TODAY

Display PMDF's count of the number of messages processed so far today.

SYNTAX

COUNTERS TODAY

Command Qualifiers	Defaults
None.	*None.*

DESCRIPTION

PMDF's count of the number of messages processed so far today may be displayed with the COUNTERS TODAY command.

EXAMPLES

This example illustrates displaying PMDF's count of the number of messages processed so far today.

```
qm.maint> COUNTERS TODAY
4263 messages processed so far today
30000 messages per day are permitted by your license
qm.maint>
```

DATE

Show the current date and time.

SYNTAX

DATE

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

None.

DESCRIPTION

The DATE command may be used to show the current date and time, in RFC 822 and RFC 1123 format. It is useful for placing time stamps in log files for command procedures which periodically run PMDF QM to check on PMDF's channel queues.

EXAMPLES

```
qm.maint> DATE
Fri, 15 Nov 2012 13:34:16 PST
qm.maint>
```

DELETE

Delete one or more messages from the channel queue directory.

SYNTAX

DELETE *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*
/LOG	*/LOG*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification number or numbers shown by a previous DIRECTORY command. Ranges are allowed.

DESCRIPTION

The DELETE command is used to delete one or more messages from the channel queue directories. The messages to be deleted are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

Note that the DELETE command irrevocably deletes each message it is instructed to delete: the messages are not returned to their originators nor will any further attempts to be made to deliver them to their recipients. The messages are permanently deleted. Often, it is preferable to use the RETURN command so as to return the message to its originator, (*e.g.,* bounce it back to the sender).

QUALIFIERS

/ALL
/NOALL (default)
Delete all messages shown by the last DIRECTORY command. When used in conjunction with the /CHANNEL qualifier, only those messages shown by the last DIRECTORY command for the specified channel will be deleted.

Unless /NOCONFIRM is specified with /ALL, you will be required to confirm any DELETE/ALL operation.

/CHANNEL=name
Specifies the name of the channel from which to delete messages. Wildcards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will be prompted to confirm each message delete operation.

/LOG (default)
/NOLOG
Specifies whether informational messages for each message delete operation are generated.

EXAMPLES

In the following example, the DIRECTORY command is used to list the messages in the local, l, channel. Then, the DELETE command is used to delete messages 1, 3, 20, 21, and 22. A range specification, 20-22, is used to specify message numbers 20, 21, and 22.

```
qm.maint> DIRECTORY L
Mon, 23 Sep 2012 13:43:39 PDT
Data gathered from the queue directory tree

Channel: l                          Size Queued since
------------------------------------------------------------
    1 ZZ01HNP17LSUWY9D4DNR.00          4 23-SEP-2012 01:10:23
    2 ZZ01HNP1RP3B6G9D4DNR.00         10 23-SEP-2012 01:10:24
    3 ZZ01HNP42MAMAI9D4DNR.00          3 23-SEP-2012 01:10:24
    4 ZZ01HNP4MEWC8G9D4DNR.00          8 23-SEP-2012 06:18:57
      ...
   24 ZZ01HNP90X63ZG9D4DNR.00          6 23-SEP-2012 13:21:14
------------------------------------------------------------
Total size:                         108

24 total messages queued
qm.maint> DELETE 1,3,20-22
%QM-I-DELETED, deleted the message file PMDF_QUEUE:[L]ZZ01HNP17LSUWY9D4DNR.00
%QM-I-DELETED, deleted the message file PMDF_QUEUE:[L]ZZ01HNP42MAMAI9D4DNR.00
%QM-I-DELETED, deleted the message file PMDF_QUEUE:[L]ZZ01HNP76RTGHY9D4DNR.00
%QM-I-DELETED, deleted the message file PMDF_QUEUE:[L]ZZ01HNP82HTXYB9D4DNR.00
%QM-I-DELETED, deleted the message file PMDF_QUEUE:[L]ZZ01HNP83JPOCV9D4DNR.00
qm.maint>
```

DIRECTORY

List currently queued messages.

SYNTAX

DIRECTORY *[channel-name]*

Command Qualifiers	Defaults
/DATABASE	*See text*
/DIRECTORY_TREE	*See text*
/ENVELOPE	*/NOENVELOPE*
/FILE_INFO	*/FILE_INFO*
/FROM	*See text*
/HELD	*/NOHELD*
/MATCH	*See text*
/OWNER	*See text*
/TO	*See text*
/TOTAL	*See text*

PARAMETERS

channel-name
An optional parameter specifying the channel for which to obtain a directory listing. Wildcards are permitted.

DESCRIPTION

The DIRECTORY command is used to show the currently queued message files in either all channel queues or a particular channel queue. In the listing, message identification numbers will appear to the left of each message file name. These numbers may be used with the DELETE, HISTORY, HOLD, READ, RELEASE, and RETURN commands so as to identify which message to operate on.

The DIRECTORY command produces its listing by looking at either the actual queue directory tree on disk, or by looking at the queue cache database. Use either the VIEW command or the /DIRECTORY_TREE or /DATABASE qualifiers to control the source of information used. Note that when /DIRECTORY_TREE or VIEW DIRECTORY_TREE is used, the “queued since” dates are the date and time that the message file was created; when /DATABASE or VIEW DATABASE is used, the queued since dates are the date and time that the message was enqueued and may pre-date the actual creation date for the message file itself.

QUALIFIERS

/DATABASE
/DIRECTORY_TREE
Controls whether the information presented is gathered from the queue cache database, /DATABASE, or by looking at the actual directory tree containing the channel queues, /DIRECTORY_TREE.

When neither /DATABASE nor /DIRECTORY_TREE is specified, then the "view" selected with the VIEW command will be used. If no VIEW command has been issued, then /DIRECTORY_TREE is assumed.

/ENVELOPE
/NOENVELOPE (default)
Use the /ENVELOPE qualifier to generate a directory listing including the envelope From: address and the list of envelope To: recipients for each listed message. By default, envelope information is not displayed as it involves opening each message file and reading through its envelope.

/FILE_INFO (default)
/NOFILE_INFO
By default, message file size and creation date information is gathered. However, this requires accessing each message file. Specify /NOFILE_INFO if you want to avoid that overhead.

/FROM=address
This qualifier may be used to request showing only those messages with the specified envelope From: address. This qualifier implies /ENVELOPE. To specify an empty (blank) envelope From: address, use /FROM=<>.

/HELD
/NOHELD (default)
Show information only for those channels with held messages.

/MATCH=keyword
This qualifier controls the interpretation of the /FROM and /TO qualifiers. Valid keywords are AND and OR.

/OWNER=username
This qualifier may be used to request showing only those message "owned" by the specified username. This qualifier implies /DATABASE. Note that messages submitted via SMTP with authentication (SMTP AUTH) will be considered to be owned by the username that authenticated, prefixed with the asterisk, `*`, character. For instance, if user JDOE submits a message from an IMAP client that successfully performs SMTP authentication, then PMDF QM will consider the owner of the message to be `*JDOE`, and to see such messages one would use the command

```
qm.maint> DIR/OWNER=*JDOE
```

/TO=address
This qualifier may be used to request showing only those messages with the specified envelope To: address. This qualifier implies /ENVELOPE.

/TOTAL
This qualifier may be used to request showing only the total number of messages, rather than listing each individual message as is the default.

EXAMPLES

1
```
qm.maint> DIRECTORY *TCP_*
Mon, 23 Sep 2012 14:53:39 PST
Data gathered from the queue directory tree

Channel: tcp_local               Size Queued since
-------------------------------------------------------------
    1 ZL01HNM78RMBP496VPJS.00        4 21-SEP-2012 09:12:29.53
    2 ZM01HNMEDX5T8E96VQDN.00       10 21-SEP-2012 12:36:41.35
    3 ZX01HNP9IO1ZAM96W55R.00        6 21-SEP-2012 13:50:06.89
    4 ZY01HNP9HTAO9696W55R.00        5 21-SEP-2012 13:49:25.61
    5 ZY01HNPBGF8JVI96W55R.00        6 21-SEP-2012 14:45:34.33
    6 ZZ01HNPBFPQ4LG96W55R.00        5 21-SEP-2012 14:45:00.01
    7 ZZ01HNPBFQ4BS896W55R.00        5 21-SEP-2012 14:45:00.53
    8 ZZ01HNPBFR5KG296W55R.00        5 21-SEP-2012 14:45:01.92
    9 ZZ01HNPBFRD2IC96W55R.00        5 21-SEP-2012 14:45:02.19
   10 ZZ01HNPBFS7VP896W55R.00        5 21-SEP-2012 14:45:03.36
   11 ZZ01HNPBFTM8YY96W55R.00        5 21-SEP-2012 14:45:05.23
   12 ZZ01HNPBFY7JYU96W55R.00        5 21-SEP-2012 14:45:11.41
   13 ZZ01HNPBGL2BYC96W55R.00        5 21-SEP-2012 14:45:42.10
-------------------------------------------------------------
Total size:                         71

Channel: mtcp_gateway             Size Queued since
-------------------------------------------------------------
    1 ZY01HNP9HYJ0QK96W55R.00        6 23-SEP-2012 13:49:32.60
    2 ZY01HNP9ID452296W55R.00        6 23-SEP-2012 13:49:52.18
    3 ZZ01HNPBFT1MAC96W55R.00        5 23-SEP-2012 14:45:04.47
    4 ZZ01HNPBGH5OAM96W55R.00        5 23-SEP-2012 14:45:36.85
    5 ZZ01HNPBGZO97C96W55R.00        5 23-SEP-2012 14:46:01.73
-------------------------------------------------------------
Total size:                         27

Grand total size:                   98
28 total messages queued
qm.maint>
```

This example shows how to use the DIRECTORY command to list the messages queued to all channels whose names match the pattern "*tcp_*"; *i.e.,* all TCP/IP channels.

2
```
qm.maint> DIRECTORY/HELD
Mon, 23 Sep 2012 13:45:18 PST
Data gathered from the queue directory tree

Channel: tcp_local               Size Queued since
-------------------------------------------------------------
    1 ZZG01HNM78RMBP496VPJS.HELD    10 12-SEP-2012 23:31:18.34
    2 ZZM01HNMEDX5T8E96VQDN.HELD     8  8-JUL-2012 13:36:14.89
    3 ZZX01HNP9IO1ZAM96W55R.HELD    23 29-AUG-2012 07:27:49.01
-------------------------------------------------------------
Total size:                         41
```

```
Grand total size:                    41
3 total held messages queued
qm.maint>
```

In this example, the /HELD qualifier is used to check for held messages.

EDIT_FAX

Edit a queued PMDF-FAX message.

SYNTAX

EDIT_FAX *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*
/LOG	*/LOG*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous DIRECTORY command. Ranges are allowed.

DESCRIPTION

The addresses of queued FAX messages may be edited so as, for instance, to correct an incorrect FAX telephone number. The messages to be edited are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

QUALIFIERS

/ALL
/NOALL (default)
Edit all messages shown by the last DIRECTORY command. When used in conjunction with the /CHANNEL qualifier, only those messages shown by the last DIRECTORY command for the specified channel will be edited.

Unless /NOCONFIRM is specified with /ALL, you will be required to confirm any EDIT_FAX/ALL operation.

/CHANNEL=name
Specifies the name of the channel from which to edit messages. Wildcards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will prompted to confirm each message edit operation.

/LOG (default)
/NOLOG
Specifies whether informational messages for each message edit operation are generated.

EXIT

Exit the PMDF QM utility.

SYNTAX **EXIT**

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS *None.*

DESCRIPTION

The EXIT and QUIT commands exit the PMDF QM utility.

HELD

List currently queued messages which have been marked as held.

SYNTAX

HELD *[channel-name]*

Command Qualifiers	Defaults
/DATABASE	*See text*
/DIRECTORY_TREE	*See text*
/ENVELOPE	*See text*
/FILE_INFO	*/FILE_INFO*
/HELD	*/HELD*

PARAMETERS

channel-name
An optional parameter specifying the channel for which to obtain a directory listing. Wildcards are permitted.

DESCRIPTION

The HELD command is a synonym for the DIRECTORY/HELD command. See the description of the DIRECTORY command for further information.

QUALIFIERS

/DATABASE
/DIRECTORY_TREE
Controls whether the information presented is gathered from the queue cache database, /DATABASE, or by looking at the actual directory tree containing the channel queues, /DIRECTORY_TREE.

When neither /DATABASE or /DIRECTORY_TREE is specified, then the "view" selected with the VIEW command will be used. If no VIEW command has been issued, then /DIRECTORY_TREE is assumed.

/ENVELOPE
Display envelope To: and From: for the held messages listed.

/FILE_INFO
/NOFILE_INFO (default)
By default, message file size and creation date information is gathered. However, this requires accessing each message file. Specify /NOFILE_INFO if you want to avoid that overhead.

/HELD (default)
/NOHELD
Show information only for those channels with held messages.

HELP

Obtain help on the use of PMDF QM.

SYNTAX

HELP *[topic]*

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

topic
Optional topic to obtain help on.

DESCRIPTION

The HELP command may be used to obtain information on PMDF QM commands. To obtain information on all of the PMDF QM commands, use the command

```
qm.maint> HELP
```

To obtain information on individual commands or topics use the command

```
qm.maint> HELP topic
```

where *topic* is the name of the command or topic of interest.

HISTORY

Display message history information.

SYNTAX

HISTORY *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous DIRECTORY command. Ranges are allowed.

DESCRIPTION

For many channels, delivery history information is appended to the end of each message file after an unsuccessful delivery attempt has been made. With the HISTORY command, this information can be displayed.

The messages to show histories for are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

Note that history information is not recorded by some channels.

QUALIFIERS

/ALL
/NOALL (default)
Display history information for all messages shown with the last DIRECTORY command. When used in conjunction with the /CHANNEL qualifier, only histories of those messages shown with the last DIRECTORY command for the specified channel will be shown.

/CHANNEL=name
Specifies the name of the channel for which to show message histories. Wild cards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will be prompted to confirm whether or not to display the history for each selected message.

HOLD

Mark one or more messages as being held.

SYNTAX

HOLD *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*
/LOG	*/LOG*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous DIRECTORY command. Ranges are allowed.

DESCRIPTION

Use the HOLD command to mark as held any messages which should temporarily be placed on "hold". PMDF will not attempt to deliver any messages which are marked as held. To resume processing of a held message, use the RELEASE command. Messages which have been held can be listed with the DIRECTORY/HELD command.

The messages to be held are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

QUALIFIERS

/ALL
/NOALL (default)
Hold all messages shown by the last DIRECTORY command. When used in conjunction with the /CHANNEL qualifier, only those messages shown by the last directory command for the specified channel will be held.

Unless /NOCONFIRM is specified with /ALL, you will be required to confirm any HOLD/ALL operation.

/CHANNEL=name
Specifies the name of the channel from which to hold messages. Wildcards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will be prompted to confirm each message hold operation.

/LOG (default)
/NOLOG
Specifies whether informational messages for each message hold operation are generated.

EXAMPLES

In the following example, the DIRECTORY command is used to list the messages in the local, l, channel. Then, the HOLD command is used to hold messages 1, 3, 20, 21, and 22. A range specification, 20-22, is used to specify message numbers 20, 21, and 22.

```
qm.maint> DIRECTORY L
Fri, 15 Nov 2012 13:43:39 PDT
Data gathered from the queue directory tree

Channel: l                          Size Queued since
------------------------------------------------------------
    1 ZZ01HNP17LSUWY9D4DNR.00          4 15-NOV-2012 01:10:23
    2 ZZ01HNP1RP3B6G9D4DNR.00         10 15-NOV-2012 01:10:24
    3 ZZ01HNP42MAMAI9D4DNR.00          3 15-NOV-2012 01:10:24
    4 ZZ01HNP4MEWC8G9D4DNR.00          8 15-NOV-2012 06:18:57
      ...
   24 ZZ01HNP90X63ZG9D4DNR.00          6 15-NOV-2012 13:21:14
------------------------------------------------------------

24 total messages queued
qm.maint> HOLD 1,3,20-22
%QM-I-HELD, held the message file PMDF_QUEUE:[L]ZZ01HNP17LSUWY9D4DNR.00
%QM-I-HELD, held the message file PMDF_QUEUE:[L]ZZ01HNP42MAMAI9D4DNR.00
%QM-I-HELD, held the message file PMDF_QUEUE:[L]ZZ01HNP76RTGHY9D4DNR.00
%QM-I-HELD, held the message file PMDF_QUEUE:[L]ZZ01HNP82HTXYB9D4DNR.00
%QM-I-HELD, held the message file PMDF_QUEUE:[L]ZZ01HNP83JPOCV9D4DNR.00
qm.maint>
```

QUIT

Exit the PMDF QM utility.

SYNTAX **QUIT**

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS *None.*

DESCRIPTION

The EXIT and QUIT commands exit the PMDF QM utility.

READ

Display message envelope and header information.

SYNTAX

READ *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*
/CONTENT	*/NOCONTENT*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous DIRECTORY command. Ranges are allowed.

DESCRIPTION

The READ command may be used to display envelope and header information for one or more queued messages. To also view the message content, use the /CONTENT qualifier.

The messages to display are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

QUALIFIERS

/ALL
/NOALL (default)
Display all messages shown with the last DIRECTORY command. When used in conjunction with the /CHANNEL qualifier, only those messages shown with the last DIRECTORY command for the specified channel will be shown.

/CHANNEL=name
Specifies the name of the channel from which to display messages. Wildcards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will be prompted to confirm whether or not to display each selected message.

/CONTENT
/NOCONTENT (default)
When /CONTENT is specified, the content of the message will also be shown.

EXAMPLES

In the following example, the envelope and header information for message number 1 is displayed.

```
qm.maint> READ 1
Filename: PMDF_QUEUE:[L]ZZ01HNPFR2FUN89D4GAS.00

Message id: 1
Transport layer information:
------------------------------------------------------------------------
Envelope From: address: fresnel@example.com
Envelope To: addresses: bernoulli

Message header:
------------------------------------------------------------------------
Received: from EXAMPLE.COM by EXAMPLE.COM (PMDF V6.1-1 #8790)
 id <01HNPFR0P5OW9D4GAS@EXAMPLE.COM> for BERNOULLI@EXAMPLE.COM; Fri,
 15 Nov 2012 16:48:41 -0700 (PDT)
Date: Fri, 15 Nov 2012 16:48:40 -0700 (PDT)
From: Fresnel the tabby cat <fresnel@example.com>
To: bernoulli@example.com
Subject: catnip and catnaps
Message-id: <01HNPFR12JYA9D4GAS@EXAMPLE.COM>
MIME-version: 1.0
Content-type: TEXT/PLAIN; CHARSET=US-ASCII
Content-transfer-encoding: 7BIT

qm.maint>
```

RELEASE

Release one or more held messages.

SYNTAX

RELEASE *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*
/LOG	*/LOG*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous DIRECTORY/HELD command. Ranges are allowed.

DESCRIPTION

Use the RELEASE command to release any messages previously marked as held, re-enter them in the queue cache database, and run the associated channel so the messages can be processed. Messages which have been held can be listed with the DIRECTORY/HELD command.

The messages to be released are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

QUALIFIERS

/ALL
/NOALL (default)
Release all messages shown by the last DIRECTORY/HELD command. When used in conjunction with the /CHANNEL qualifier, only those messages shown by the last DIRECTORY/HELD command for the specified channel will be released.

Unless /NOCONFIRM is specified with /ALL, you will be required to confirm any RELEASE/ALL operation.

/CHANNEL=name
Specifies the name of the channel from which to release messages. Wildcards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will be prompted to confirm each message release operation.

/LOG (default)
/NOLOG
Specifies whether informational messages for each message release operation are generated.

EXAMPLES

In the following example, the DIRECTORY/HELD command is used to list held messages in the tcp_local channel. Then, the RELEASE command is used to release all of the held messages from that channel.

```
qm.maint> DIRECTORY/HELD TCP_LOCAL
Fri, 10 Mar 2012 13:43:39 PDT
Data gathered from the queue directory tree

Channel: tcp_local                Size Queued since
------------------------------------------------------------
    1 ZZ01HNP17LSUWY9D4DNR.HELD      4 10-MAR-2012 03:12:00
    2 ZZ01HNP1RP3B6G9D4DNR.HELD     10 10-MAR-2012 11:46:23
    3 ZZ01HNP42MAMAI9D4DNR.HELD      5 11-MAR-2012 18:17:01
------------------------------------------------------------
Total size:                       19

3 total messages queued
qm.maint> RELEASE/ALL
Release all message files (Y/N, default is N)? YES
%QM-I-RELEASED, released the message file
   PMDF_QUEUE:[TCP_LOCAL]ZZ01HNP17LSUWY9D4DNR.HELD
%QM-I-RELEASED, released the message file
   PMDF_QUEUE:[TCP_LOCAL]ZZ01HNP1RP3B6G9D4DNR.HELD
%QM-I-RELEASED, released the message file
   PMDF_QUEUE:[TCP_LOCAL]ZZ01HNP42MAMAI9D4DNR.HELD
qm.maint>
```

RETURN

Return a message to its sender.

SYNTAX

RETURN *[message-id[,...]]*

Command Qualifiers	Defaults
/ALL	*/NOALL*
/CHANNEL=name	*None*
/CONFIRM	*/NOCONFIRM*
/LOG	*/LOG*

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous DIRECTORY command. Ranges are allowed.

DESCRIPTION

Queued messages may be returned to their originator with the RETURN command. The messages to be returned are specified by their message identification numbers shown by the most recent DIRECTORY command. That number appears in the leftmost column of the DIRECTORY command listing. Ambiguous message numbers must be qualified by the proper channel name with the /CHANNEL qualifier.

The returned message is in two parts. The first part explains the reason why the message is being returned; the text of the reason is contained in the file `return_bounced.txt` file located in the PMDF language-specific directory. The second part of the returned message contains the original message itself.

QUALIFIERS

/ALL
/NOALL (default)
Return all messages shown by the last DIRECTORY command. When used in conjunction with the /CHANNEL qualifier, only those messages shown by the last DIRECTORY command for the specified channel will be returned.

Unless /NOCONFIRM is specified with /ALL, you will be required to confirm any RETURN/ALL operation.

/CHANNEL=name
Specifies the name of the channel from which to return messages. Wildcards are not permitted.

/CONFIRM
/NOCONFIRM (default)
When /CONFIRM is specified, you will be prompted to confirm each message return operation.

/LOG (default)
/NOLOG
Specifies whether informational messages for each message return operation are generated.

SPAWN

Create a subprocess.

SYNTAX

SPAWN *[command]*

Command Qualifiers	Defaults
/INPUT=in-file-spec	*None*
/LOGICAL_NAMES	*/LOGICAL_NAMES*
/OUTPUT=out-file-spec	*None*
/PROCESS=name	*None*
/SYMBOLS	*/SYMBOLS*
/WAIT	*/WAIT*

restrictions Cannot be used from a captive account.

PARAMETERS

command
Optional parameter specifying the command string for the subprocess to execute. After the command completes, the subprocess terminates and control is returned to the parent process.

DESCRIPTION

The SPAWN command may be used to either issue a single DCL command from within PMDF QM or to leave PMDF QM temporarily, do other work (*e.g.,* type out a file, generate a directory listing, *etc.*), and then return to PMDF QM.

By default, the context of the current process is copied to the subprocess. This behavior may be controlled with the /LOGICAL_NAMES and /SYMBOLS qualifiers.

QUALIFIERS

/INPUT=in-file-spec
Specifies an input command file from which the subprocess is to draw command input. Once command processing is completed, the subprocess terminates. When you specify both a command string and input file, then the command string is first processed and then the commands from the input file.

/LOGICAL_NAMES (default)
/NOLOGICAL_NAMES
The /LOGICAL_NAMES qualifier specifies that the logical names of the parent process are to be copied to the subprocess. This is the default behavior. Specify /NOLOGICAL_NAMES to prevent the subprocess from inheriting the logical name definitions of its parent.

/OUTPUT=out-file-spec
Specifies the output file to which the output of the subprocess is to be directed. If the /OUTPUT qualifier is omitted, then subprocess output is directed to the current SYS$OUTPUT device (generally, your terminal).

/PROCESS=name
Specifies the process name to associate with the subprocess. If not specified, a default name of the form USERNAME_n, where "USERNAME" is your username, is used.

/SYMBOLS (default)
/NOSYMBOLS
The /SYMBOLS qualifier specifies that the DCL symbol definitions of the parent process are to be copied to the subprocess. This is the default behavior. Specify /NOSYMBOLS to prevent the subprocess from inheriting the symbol definitions of its parent.

/WAIT (default)
/NOWAIT
By default, your current (parent) process will wait until the subprocess has finished its processing and terminated. This default behavior is explicitly selected with the /WAIT qualifier. The /NOWAIT qualifier allows you to continue working from your current process while the subprocess is running. When you specify /NOWAIT, you should also specify the /OUTPUT qualifier so as to prevent the subprocess output from appearing on your terminal screen.

EXAMPLES

1
```
qm.maint> SPAWN DIRECTORY/SIZE=ALL a.txt

Directory D1:[BOB]

A.TXT;10    125/126
A.TXT;9     124/126
A.TXT;8     124/126

Total of 3 files, 373/378.
qm.maint> SPAWN PURGE/LOG a.txt
%PURGE-I-FILPURG, D1:[BOB]A.TXT;9 deleted (126 blocks)
%PURGE-I-FILPURG, D1:[BOB]A.TXT;8 deleted (126 blocks)
%PURGE-I-TOTAL, 2 files deleted (252 blocks)
qm.maint>
```

In this example, the SPAWN command is used to obtain a directory listing of the files `a.txt`, and then to purge back old versions of that file. The ability to do this is useful when you find that you have insufficient disk quota to create and edit a mail message you want to send.

```
2   qm.maint> SPAWN
       .
       .
       .
    $ LOGOUT
      Process BOB_1 logged out at 15-NOV-2012 12:12:51.42
    qm.maint>
```

In this example a SPAWN command with no command string is issued. This places you into the subprocess where you can issue DCL commands and perform other processing. When you are done with the subprocess and ready to return to PMDF QM, use the LOGOUT or EOJ command.

SUMMARIZE

Display a summary listing of message files.

SYNTAX

SUMMARIZE

Command Qualifiers	Defaults
/DATABASE	*See text*
/DIRECTORY_TREE	*See text*
/HEADING	*/HEADING*
/HELD	*/NOHELD*
/TRAILING	*/TRAILING*

PARAMETERS

None.

DESCRIPTION

Display a summary listing of message files.

COMMAND QUALIFIERS

/DATABASE
/DIRECTORY_TREE
Controls whether the information presented is gathered from the queue cache database, /DATABASE, or by looking at the actual directory tree containing the channel queues, /DIRECTORY_TREE.

When neither /DATABASE or /DIRECTORY_TREE is specified, then the "view" selected with the VIEW command will be used. If no VIEW command has been issued, then /DIRECTORY_TREE is assumed.

/HEADING (default)
/NOHEADING
Controls whether or not a heading line describing each column of output is displayed at the start of the summary listing.

/HELD
/NOHELD (default)
Controls whether or not to include counts of `.HELD` messages in the output.

/TRAILING (default)
/NOTRAILING
Controls whether or not a trailing line with totals is displayed at the end of the summary.

EXAMPLES

The following example shows displaying a summary listing of message files.

```
qm.maint> SUMMARIZE
                                                Messages
                          Channel  Queued   Size (Kb)   Oldest
-------------------------------- -------- ----------- -----------------
                        cc_local       0        0.00
                     circuitcheck      4        7.51    8 Jun, 10:19:20
                       conversion      0        0.00
                                l      0        0.00
                         mailserv      0        0.00
                     mime_to_x400      0        0.00
                         mr_local      0        0.00
                         popstore      0        0.00
                          process      0        0.00
                        reprocess      0        0.00
                     tcp_internal     15       51.47    2 Jun, 12:10:03
                        tcp_local      0        0.00
                        wpo_local      0        0.00
                       x400_local      0        0.00
                     x400_to_mime      0        0.00
-------------------------------- -------- ----------- -----------------
                           Totals     19       58.98

qm.maint>
```

TOP

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues.

SYNTAX

TOP *[channel]*

Command Qualifiers	Defaults
/CONTENT[=offset-specifier]	*None*
/DATABASE	*See text*
/DIRECTORY_TREE	*See text*
/ENV_FROM[=offset-specifier]	*None*
/MIN_COUNT=n	*/MIN_COUNT=2*
/SUBJECT[=offset-specifier]	*/SUBJECT=(START=1,LENGTH=2147483647)*
/THREADS=n	*/NOTHREADS*
/TOP=n	*/TOP=20*
/VERBOSE	*/NOVERBOSE*

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be scanned for string frequencies. * or ? wildcard characters may be used in the channel specification.

DESCRIPTION

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues. By default, only Subject: fields are shown (/SUBJECT). Use /ENV_FROM to display frequent envelope From: fields or /CONTENT to display frequent message contents. Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used.

The optional channel parameter restricts the scan to message files in the specified channel. The channel parameter may use * and ? wild cards.

By default, the top 20 most frequently occurring fields are shown (/TOP=20) provided that they occur 2 or more times (/MIN_COUNT=2). Use the /TOP and /MIN_COUNT qualifiers to alter this behavior. The message files searched may be either all those present in the channel queue directory tree, or only those files with entries in the queue cache database. Use either the VIEW command of the /DIRECTORY_TREE or /DATABASE qualifier to control which files are searched.

The /THREADS qualifier may be used to accelerate scanning on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous scanning threads, specify /THREADS=*n*. The value n must be in the range 1-8. The default is /NOTHREADS.

The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers accept the optional qualifiers START=n and LENGTH=n. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are /CONTENT=(START=1,LENGTH=256), /ENV_FROM=(START=1,LENGTH=2147483647), and /SUBJECT=(START=1,LENGTH=2147483647). Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

COMMAND QUALIFIERS

/CONTENT[=offset-specifier]
/ENV_FROM[=offset-specifier]
/SUBJECT[=offset-specifier]
The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (/SUBJECT). Use /ENV_FROM to display frequent envelope From: fields or /CONTENT to display frequent message contents. Any combination of /CONTENT, /ENV_FROM, and /SUBJECT may be specified. However, only one of each may be used.

The /CONTENT, /ENV_FROM, and /SUBJECT qualifiers accept the optional qualifiers START=n and LENGTH=n. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are /CONTENT=(START=1,LENGTH=256), /ENV_FROM=(START=1,LENGTH=2147483647), and /SUBJECT=(START=1,LENGTH=2147483647). Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

/DATABASE
/DIRECTORY_TREE
Controls whether the message files scanned are only those with entries in the queue cache database, /DATABASE, or all message files actually present in the channel queue directory tree, /DIRECTORY_TREE.

When neither /DATABASE nor /DIRECTORY_TREE is specified, then the "view" selected with the VIEW command will be used. If no VIEW command has been issued, then /DIRECTORY_TREE is assumed.

/MIN_COUNT=n
By default, a string must occur at least 2 times, /MIN_COUNT=2, in order to be displayed.

/THREADS=n
/NOTHREADS (default)
The /THREADS qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify /THREADS=*n*. The value *n* must be an integer in the range 1-8. The default is /NOTHREADS.

/TOP=n
By default, the top 20 most frequently occurring fields are shown, (/TOP=20).

/VERBOSE
/NOVERBOSE (default)
The /VERBOSE qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

The following example shows displaying the most frequently occurring Subject: and envelope From: addresses amongst messages in the PMDF queue area.

```
qm.maint> TOP/SUBJECT/ENV_FROM
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Envelope From: addresses which occur 2 or more times
 Count  Envelope From: address
======  ============================================================================
    27
    10  owner-ex-list@example.com
     2  owner-test-list@example.com

Top 20 Subject: header lines which occur 2 or more times
 Count  Subject
======  ============================================================================
     6  Re: your ex-list posting
     2  Test posting to test-list
```

The following example shows displaying the most frequently occuring Subject: lines that occur 20 times or more, starting from 12 characters into the Subject: header value. This may be useful when trying to spot spam that inserts random characters at the beginning of the Subject: header value.

```
qm.maint> TOP/SUBJECT=START=12/MIN_COUNT=15
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Subject: header lines which occur 15 or more times
 Count  Subject
======  ============================================================================
    25  ake money fast $$$
```

VIEW

Control whether the DIRECTORY command shows the channel queue directory tree or the queue cache database.

SYNTAX

VIEW *type*

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

type
The type of view to use: DIRECTORY_TREE or DATABASE

DESCRIPTION

The DIRECTORY command produces its listing by looking at either the actual channel queue directory tree on disk, or by looking at the queue cache database. The VIEW command controls which is used. By default, the view is the channel queue directory tree. Issue the command,

```
qm.maint> VIEW DATABASE
qm.maint>
```

to switch to viewing the queue cache database. The command

```
qm.maint> VIEW DIRECTORY_TREE
qm.maint>
```

will switch you back to viewing the channel queue directory tree. Issuing the VIEW command without any parameter will restore the default behavior and is thus equivalent to the VIEW DIRECTORY_TREE command.

30 Utilities on UNIX

PMDF contains a modest collection of management utility programs, which are used to perform various maintenance, testing, and management tasks. The following sections describe these utilities. Note that many of the utilities are mentioned elsewhere in this document in the context of how they are actually used. User-level utilities are described in the *PMDF User's Guide*.

Briefly, the PMDF utilities, both those documented in the *PMDF User's Guide* and those documented here, are shown in Table 30–1. Those utilities only available under OpenVMS are marked with a †; those only available under UNIX are marked with a ‡.

Table 30–1 PMDF Utilities

Web-based utilities and displays	
URL	**Description**
`http://`*`pmdfhost`*`:7633/configure/`	Configure: generate PMDF configuration files; see the *PMDF Installation Guide*
`http://`*`pmdfhost`*`:7633/dispatcher/`	Dispatcher Statistics: view statistics on recent connections to the Dispatcher, *e.g.*, SMTP, POP and IMAP connections
`http://`*`pmdfhost`*`:7633/mailbox_filters/`	Mailbox Filters: generate and modify system and user mailbox filters controlling filtering of incoming messages
`http://`*`pmdfhost`*`:7633/qm/`	Message Queue Management: queue management utility
`http://`*`pmdfhost`*`:7633/msgstore/`	MessageStore Administration: manage PMDF MessageStore; see the *PMDF popstore & MessageStore Manager's Guide*
`http://`*`pmdfhost`*`:7633/monitor/`	Monitoring: view PMDF counters; on OpenVMS, also view the status of PMDF processing queues
`http://`*`pmdfhost`*`:7633/chng_pwd/`	Password Change Utility: change your e-mail password; usually used to change a PMDF MessageStore or PMDF popstore account password, but may also change a system password, depending upon configuration; see the *PMDF popstore & MessageStore Manager's Guide*

Table 30–1 (Cont.) PMDF Utilities

Web-based utilities and displays	
URL	**Description**
`http://`*`pmdfhost`*`:7633/popstore/`	popstore Administration: manage PMDF popstore; see the *PMDF popstore & MessageStore Manager's Guide*
`http://`*`pmdfhost`*`:7633/msps_user/`	popstore and MessageStore User Interface: change your password or view your account settings, or for popstore only view your messages; see the *PMDF popstore & MessageStore Manager's Guide*

	Command utilities			
	OpenVMS utility		**UNIX utility**	**Description**
†	`CACHE/CLOSE`			Have detached processes close their connections to the queue cache database
†	`CACHE/REBUILD`			Build a new, synchronized queue cache database
	`CACHE/SYNCHRONIZE`		`cache -synchronize`	Synchronize the current queue cache database
		‡	`cache -view`	View entries in the queue cache database
	`CHBUILD`		`chbuild`	Compile the PMDF character set conversion tables
	`CLBUILD`		`clbuild`	Compile a PMDF command definition file
	`CNBUILD`		`cnbuild`	Compile the PMDF configuration, alias, mapping, security, system wide filter, circuit check, and option files
	`CONFIGURE`	§	`configure`	Create a PMDF configuration file
†	`convert_cache.com`			Perform a CONVERT/RECLAIM on the queue cache
		§	`convertdb`	Read entries from a V6.0-V6.4 PMDF `crdb` database and write out a corresponding V6.5 or later PMDF `crdb` database
	`COUNTERS/CLEAR`		`counters -clear`	Clear the in-memory cache of channel counters
†	`COUNTERS/CRDB`			Create a database of channel counters

§Not available on NT

†Available on OpenVMS only

‡Available on UNIX only

Table 30–1 (Cont.) PMDF Utilities

	Command utilities		
	OpenVMS utility	**UNIX utility**	**Description**
	COUNTERS/SHOW	counters –show	Display the contents of the database of channel counters
†	COUNTERS/SYNCHRONIZE		Synchronize the in-memory cache of channel counters with the database
	COUNTERS/TODAY	counters –today	Display PMDF's count of the number of messages processed today
	CRDB	crdb	Create a PMDF database
	DB	db	Manage a personal alias database; see the *PMDF User's Guide*
†	DCF		Convert WPS and DX files to ASCII; provided with PMDF-MR
	DECODE	decode	Decode a file encoded using MIME encodings; see the *PMDF User's Guide*
™	DUMPDB	dumpdb	Dump entries in a PMDF crdb database to a flat text file
	®	edit	Edit PMDF configuration files
	ENCODE	encode	Encode a file using MIME encodings; see the *PMDF User's Guide*
	‡	find	Find the filename corresponding to the specified "version" of a PMDF file
†	FOLDER		Place a message file into a VMS MAIL folder; see the *PMDF User's Guide, OpenVMS Edition*
†	FORWARD		Set a forwarding address in the PMDF alias database; see the OpenVMS Edition of the *PMDF User's Guide*
†	G3		Analyze a PMDF-FAX G3 file; provided with PMDF-FAX
†	INSTALL		Install or deinstall PMDF images and databases
	KILL §	kill	Kill the specified PMDF component

™See also the PMDF DB (OpenVMS) or pmdf db (UNIX or NT) utility's write *filename* alias command.

®Available on NT only

§Not available on NT

†Available on OpenVMS only

‡Available on UNIX only

Table 30–1 (Cont.) PMDF Utilities

	Command utilities		
	OpenVMS utility	**UNIX utility**	**Description**
	LICENSE	license –verify	On OpenVMS, activate or deactivate PMDF bundle licenses on a node; on Solaris, Linux, and Windows, verify the validity of a PMDF license file
†	MAIL		An extended version of VMS MAIL; see the OpenVMS Edition of the *PMDF User's Guide*
	migrate	§ migrate	Copy message folders from one IMAP host to another IMAP host; see the appropriate edition of the *PMDF User's Guide*
	MOVEIN	§ movein	Migrate a user's mailbox from one message store to another; see the *PMDF popstore & MessageStore Manager's Guide*
	MSGSTORE	msgstore	Interactive PMDF Message Store management utility; see the *PMDF popstore & MessageStore Manager's Guide*
	PASSWORD	password	Set remote authentication passwords
	POPSTORE	popstore	Interactive PMDF popstore management utility; see the *PMDF popstore & MessageStore Manager's Guide*
	PROCESS	§ process	List currently running PMDF jobs
		§ profile	Set local user's choice of delivery mechanism in the PMDF user profile database
†	PS		Convert text and Runoff .mem files to PostScript; provided with PMDF-FAX; see the OpenVMS Edition of the *PMDF User's Guide*
		‡ purge	Purge PMDF log files
	QCLEAN	qclean	Hold or delete message files matching specified criteria
	QM	qm	Manage PMDF message queues; see also the web-based QM utility, Section 31.2

§Not available on NT

†Available on OpenVMS only

‡Available on UNIX only

Table 30–1 (Cont.) PMDF Utilities

Command utilities		
OpenVMS utility	**UNIX utility**	**Description**
QTOP	qtop	Display the most frequently occurring strings found in message files in the PMDF queue area
RESTART	restart	Restart detached PMDF processes
RETURN	return	Return (bounce) a mail message to its originator
master.com	run	Process messages in a specified channel
SEND	send	Send a mail message; see the *PMDF User's Guide*
SHUTDOWN	shutdown	Shut down detached PMDF processes
STARTUP	startup	Start detached PMDF processes
submit_master.com	submit	Process messages in a specified channel
submit_master.com	submit_master	Process messages in a specified channel—on UNIX, a synonym for submit
TEST/MAPPING	test -mapping	Test a mapping table
TEST/MATCH	test -match	
TEST/URL	test -url	Test an LDAP query URL
\\ tls_certdump	\Dump the contents of a certificate file)	
tls_certreq	tls_certreq	Generate a public key pair and a certificate request
tls_ciphers	tls_ciphers	List available ciphers
	‡ view	Display the specified "version" of a PMDF file
VERSION	version	Print PMDF version number

‡Available on UNIX only

This chapter is broken into two main sections. The first section, Command Line Utilities on UNIX, describes the command shell utilities available on UNIX; the second section, Interactive Utilities, describes the interactive pmdf profile, pmdf qm, and utilities available on UNIX.

30.1 Command Line Utilities on UNIX

This section documents the PMDF utilities available on UNIX.

On UNIX platforms, these utilities are implemented via `/usr/bin/pmdf` (which is a symbolic link to `/pmdf/bin/pmdf`). For convenient use of the `pmdf` commands, add the path `/usr/bin` to your search path.

On NT, these utilities are implemented via `\pmdf\bin` on the drive selected at installation time. The PMDF installation procedure automatically makes the path to these commands available, inserting the path under the "Start" menu, under "Settings", under "Control Panel", under "System", under "System Variables", under "Path".

cache -synchronize—Synchronize the queue cache

Update the queue cache database so as to reflect all messages currently present in the message queues.

SYNTAX

pmdf cache -synchronize

Command Qualifiers	Defaults
None.	*None.*

restrictions Must have superuser privileges (UNIX) or be logged in as Administrator (NT) in order to use this utility.

PARAMETERS

None.

DESCRIPTION

The `pmdf cache -synchronize` utility updates the active queue cache database to reflect all non-held message files currently present in the PMDF queue subdirectories, `/pmdf/queue/*` on UNIX or `C:\pmdf\queue\*` on NT.

The queue cache database consists of the files contained in the directory pointed at by PMDF_QUEUE_CACHE_DATABASE option in the PMDF tailor file, `/etc/pmdf_tailor`, on UNIX, or the PMDF_QUEUE_CACHE_DATABASE PMDF Tailor registry entry on NT. Normally, the queue cache directory is called `/pmdf/table/queue_cache` on UNIX, and is usually `C:\pmdf\table\queue_cache` (possibly on a drive other than C:) on NT. On UNIX, this directory and the files it contains should be protected against world and group access (`-rwx------`) and have the same uid as the directories `/pmdf/queue` and `/pmdf/log`.

EXAMPLES

To synchronize the queue cache, for instance after renaming a message file, issue the UNIX command

```
# pmdf cache -synchronize
```

or the NT command

```
C:\> pmdf cache -synchronize
```

cache -view—View entries in the queue cache

View the current entries for a channel in the queue cache database.

SYNTAX

pmdf cache -view *[channel-name]*

Command Qualifiers	Defaults
None.	*None.*

restrictions

Must have superuser privileges (UNIX) or be logged in as Administrator (NT) in order to use this utility.

PARAMETERS

channel-name
Optional parameter specifying the name of the channel for which to show entries. If no channel name is specified, all entries in the queue cache database will be shown.

DESCRIPTION

The `pmdf cache -view` utility shows the current entries in the PMDF queue cache database for a channel.

EXAMPLES

This UNIX example shows checking the queue cache database for entries for the tcp_local channel and finding one such entry:

```
# pmdf cache -view tcp_local
  recipient count  : 1
  subdir           : 19
  channel          : tcp_local
  filename         : ZY0ETO00805RZ35T.00
  recipient system : else.where.com
  username         : adam
  creation         : 28-May-2012 14:35:27
  expiry           :
  deferred         :
  last_try         : 28-May-2012 12:10:41
  priority         : 3
-------------------------------------------------------
```

chbuild—Character set table compiler

Compile the PMDF character set conversion tables and load the resulting image file into shared memory.

SYNTAX

pmdf chbuild

Command Qualifiers	Defaults
`-image_file=`*file-spec*	`-image_file=PMDF_CHARSET_DATA`
`-maximum`	`-nomaximum`
`-option_file=`*file-spec*	`-option_file=PMDF_CHARSET_OPTION_FILE`
`-remove`	*None*
`-sizes`	`-nosizes`
`-statistics`	`-nostatistics`

restrictions Must have superuser privileges (UNIX) or be logged in as Administrator (NT) in order to use this utility.

PARAMETERS

None.

DESCRIPTION

The `pmdf chbuild` utility compiles the character set conversion tables and loads the resulting file into shared memory.

PMDF ships with very complete character set tables so it is not normally necessary to run this utility.

COMMAND QUALIFIERS

`-image_file`*[=file-spec]*
`-noimage_file`
By default, `pmdf chbuild` creates as output the image file named by the PMDF_CHARSET_DATA option of the PMDF tailor file, `/etc/pmdf_tailor`, (UNIX) or Tailor Registry entry (NT). With the `-image_file` qualifier, an alternate file name may be specified.

When the `-noimage_file` qualifier is specified, `pmdf chbuild` does not produce an output file. This qualifier is used in conjunction with the `-option_file` qualifier to produce as output an option file which specifies table sizes adequate to hold the tables required by the processed input files.

-maximum
-nomaximum *(default)*
The file /pmdf/table/maximum_charset.dat is read in addition to the file named by the PMDF_CHARSET_OPTION_FILE option of the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or Tailor Registry entry (NT) when -maximum is specified. This file specifies near maximum table sizes but does not change any other option file parameter settings. Only use this qualifier if the current table sizes are inadequate. The -noimage_file and -option_file qualifiers should always be used in conjunction with this qualifier—it makes no sense to output the enormous configuration that is produced by -maximum, but it does make sense to use -maximum to get past size restrictions in order to build a properly sized option file so that a properly sized character set image can be built with a subsequent pmdf chbuild invocation.

-option_file*[=file-spec]*
-nooption_file
pmdf chbuild can optionally produce an option file that contains correct table sizes to hold the character set conversion tables which were just compiled (plus a little room for growth). The -option_file qualifier causes this file to be output. By default, this file is the file named by the PMDF_CHARSET_OPTION_FILE option of the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or Tailor Registry entry (NT). The value on the -option_file qualifier may be used to specify an alternate file name. If the -nooption_file qualifier is given, then no option file will be output.

pmdf chbuild always reads any option file (*i.e.*, the file named by the PMDF_OPTION_FILE option of the PMDF tailor file or Tailor key in the NT Registry) that is already present; use of this qualifier will not alter this behavior. However, use of the -maximum qualifier causes pmdf chbuild to read options from maximum_charset.dat in addition to PMDF_CHARSET_OPTION_FILE. This file specifies near maximum table sizes. Only use this qualifier if the current table sizes are inadequate, and only use it to create a new option file. The -noimage_file qualifier should always be specified when -maximum is specified since a maximum-size image would be truly enormous and extremely wasteful.

-remove
Remove any existant compiled character set conversion table; *i.e.*, remove the file named by the PMDF_CHARSET_DATA option of the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or Tailor Registry entry (NT).

-sizes
-nosizes *(default)*
The -sizes qualifier instructs pmdf chbuild to output information on the sizes of the uncompiled character set tables.

-statistics
-nostatistics *(default)*
The -statistics qualifier instructs pmdf chbuild to output information on the compiled conversion tables. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the -option_file qualifier is needed.

EXAMPLES

The standard command used on UNIX to compile character set conversion tables is:

```
# pmdf chbuild
```

or on NT:

```
C:\> pmdf chbuild
```

clbuild—Command definition compiler

Compile a PMDF command definition file and load the resulting image file into shared memory.

SYNTAX

pmdf clbuild *cld-file-spec*

Command Qualifiers	Defaults
`-debug`	`-nodebug`
`-image_file=`*file-spec*	`-noimage_file`
`-maximum`	`-nomaximum`
`-option_file=`*file-spec*	`-nooption_file`
`-remove`	*None*
`-sizes`	`-nosizes`
`-statistics`	`-nostatistics`

restrictions Must have superuser privileges (UNIX) or be logged in as Administrator (NT) in order to use this utility.

PARAMETERS

cld-file-spec
The file specification of a PMDF command definition file to read as input; *e.g.*, on UNIX `/pmdf/lib/pmdf.cld` or on NT `C:\pmdf\lib\pmdf.cld`.

DESCRIPTION

The `pmdf clbuild` utility compiles a command line definition file and loads the resulting file into shared memory.

PMDF ships with a pre-compiled command line definition image so it is not normally necessary to run this utility.

COMMAND QUALIFIERS

`-debug`
`-nodebug` ***(default)***
The `-debug` qualifier causes `pmdf clbuild` to output debug information regarding its operation.

`-image_file=`*file-spec*
`-noimage_file` ***(default)***
By default, `pmdf clbuild` does not produce a compiled command definition file. In order to produce a compiled command definition file, the file to produce must

be specified using the –image_file qualifier. Note that the PMDF tailor file option PMDF_COMMAND_DATA may be specified as the file-spec, if the goal is to produce a compiled version of the main PMDF command definition file, /pmdf/lib/pmdf.cld.

–maximum
–nomaximum ***(default)***
The file /pmdf/table/maximum_command.dat is read when –maximum is specified. This file specifies near maximum table sizes but does not change any other command option file parameter settings. Only use this qualifier if the current table sizes are inadequate. The –noimage_file and –option_file qualifiers should always be used in conjunction with this qualifier—it makes no sense to output the enormous command definition image that is produced by –maximum, but it does make sense to use –maximum to get past size restrictions in order to build a properly sized command option file so that a properly sized command definition image can be built with a subsequent pmdf clbuild invocation.

–option_file*[=file-spec]*
–nooption_file ***(default)***
pmdf clbuild can optionally produce a command option file that contains correct table sizes to hold the command definitions which were just compiled (plus a little room for growth). The –option_file qualifier causes this file to read as input and a new such option file created as output. If –option_file is specified with no value, then the file written will have the same name as the input command definition file, but with the file extension .cop; for instance, if the file /pmdf/lib/pmdf.cld was the input parameter, then the default name for the output command option file would be /pmdf/lib/pmdf.cop. If the –nooption_file qualifier is specified (the default), then no option file will be output.

Note that use of the –maximum qualifier causes pmdf clbuild to read options from maximum_command.dat in addition to any command option file. This file specifies near maximum table sizes. Only use this qualifier if the current table sizes are inadequate, and only use it to create a new option file. The –noimage_file qualifier should always be specified when –maximum is specified since a maximum-size image would be truly enormous and extremely wasteful.

–remove
Remove an existant compiled command definition image. Note that the PMDF_COMMAND_DATA name may be used to cause removal of the basic PMDF compiled command definition image, as referenced by this PMDF tailor file option (UNIX) or PMDF Tailor Registry key (NT).

–sizes
–nosizes ***(default)***
The –sizes qualifier instructs pmdf clbuild to output information on the sizes of the uncompiled command definitions.

–statistics
–nostatistics ***(default)***
The –statistics qualifier instructs pmdf clbuild to output information on the compiled command definition image. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the –option_file qualifier is needed.

EXAMPLES

The standard command used to compile the basic PMDF command definition file is on UNIX:

```
# pmdf clbuild -option_file -image_file=PMDF_COMMAND_DATA /pmdf/lib/pmdf.cld
```

or on NT:

```
C:\> pmdf clbuild -option_file -image_file=PMDF_COMMAND_DATA C:\pmdf\lib\pmdf.cld
```

cnbuild—Configuration compiler

Compile the PMDF configuration, alias, mapping, conversion, security, system wide filter, circuit check, and option files and load the resulting file into shared memory.

SYNTAX

pmdf cnbuild

Command Qualifiers	Defaults
`-image_file=`*file-spec*	`-image_file=PMDF_CONFIG_DATA`
`-maximum`	`-nomaximum`
`-option_file=`*file-spec*	`-option_file=PMDF_OPTION_FILE`
`-remove`	*None*
`-sizes`	`-nosizes`
`-statistics`	`-nostatistics`

restrictions Must have superuser privileges (UNIX) or be logged in as Administrator (NT) in order to use this utility.

PARAMETERS

None.

DESCRIPTION

The `cnbuild` utility compiles the textual configuration, option, mapping, conversion, security, system wide filter, circuit check, and alias files, and loads the resulting file into shared memory. The resulting image is the file named by the PMDF_CONFIG_DATA option of the PMDF tailor file, `/etc/pmdf_tailor`, (usually `/pmdf/lib/config_data`).

Whenever a component of PMDF (*e.g.,* a channel program) must read any possibly compiled configuration component, it first checks to see whether the file named by the PMDF tailor file option PMDF_CONFIG_DATA is loaded into shared memory; if this compiled image exists but is not loaded, PMDF loads it into shared memory. If PMDF finds (or not finding, is able to load itself) a mapped section file in shared memory, the running program uses the information in that file. There are four exceptions to this rule. The first is `pmdf cnbuild` itself, which for obvious reasons always reads the text files and never tries to use an image form of the configuration data. The remaining two exceptions are `pmdf test -rewrite` and `pmdf test -mapping`, which can all be instructed with the `-image_file` qualifier to use a different compiled configuration file. This facility in `pmdf test -rewrite` is useful for testing changes prior to compiling them.

The reason for compiling configuration information is simple: performance. The only penalty paid for compilation is the need to recompile and reload the image any time the configuration or alias files are edited. Also, be sure to restart any programs or channels which load the configuration data only once when they start up, *e.g.*, the PMDF multithreaded TCP SMTP server.

Once you begin to use a compiled configuration, it will be necessary to recompile the configuration every time changes are made to any of the following files: the PMDF configuration file (or any files referenced by it), the PMDF system alias file, the PMDF mapping file, the PMDF option file, the PMDF conversion file, the PMDF security configuration file, the system wide filter file, or the circuit check configuration file. Specifically, these are the files pointed at the PMDF tailor file options (UNIX) or PMDF Tailor Registry keys (NT) PMDF_CONFIG_FILE, PMDF_ALIAS_FILE, PMDF_MAPPING_FILE, PMDF_OPTION_FILE, PMDF_CONVERSION_FILE, and PMDF_SECURITY_CONFIG_FILE, respectively, which usually point on UNIX to

```
/pmdf/table/pmdf.cnf,
/pmdf/table/aliases,
/pmdf/table/mappings,
/pmdf/table/option.dat,
/pmdf/table/conversions, and
/pmdf/table/security.cnf,
```

or on NT to

```
C:\pmdf\table\pmdf.cnf,
C:\pmdf\table\aliases,
C:\pmdf\table\mappings,
C:\pmdf\table\option.dat,
C:\pmdf\table\conversions, and
C:\pmdf\table\security.cnf.
```

as well as the files `/pmdf/table/pmdf.filter` and `/pmdf/table/circuitcheck.cnf` (UNIX) or `C:\pmdf\table\pmdf.filter` and `C:\pmdf\table\circuitcheck.cnf` (NT). Until such time as the configuration is rebuilt, changes to any of these files will not be visible to the running PMDF system.

See Chapter 8 for further details on the use of compiled configurations.

COMMAND QUALIFIERS

`-image_file`*`[=file-spec]`*
`-noimage_file`
By default, `pmdf cnbuild` creates as output the image file named by the PMDF_CONFIG_DATA option of the PMDF tailor file, `/etc/pmdf_tailor`, (UNIX) or PMDF Tailor Registry key (NT). With the `-image_file` qualifier, an alternate file name may be specified.

When the `-noimage_file` qualifier is specified, `pmdf cnbuild` does not produce an output file. This qualifier is used in conjunction with the `-option_file` qualifier to produce as output an option file which specifies table sizes adequate to hold the configuration required by the processed input files.

–maximum
–nomaximum *(default)*
When –maximum is specified, the file /pmdf/table/maximum.dat is read in addition to the file named by the PMDF_OPTION_FILE option in the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or PMDF Tailor Registry key (NT). This file specifies near maximum table sizes but does not change any other option file parameter settings. Only use this qualifier if the current table sizes are inadequate. The –noimage_file and –option_file qualifiers should always be used in conjunction with this qualifier—it makes no sense to output the enormous configuration that is produced by –maximum, but it does make sense to use –maximum to get past size restrictions in order to build a properly sized option file so that a properly sized configuration can be built with a subsequent pmdf cnbuild invocation.

–option_file*[=file-spec]*
–nooption_file *(default)*
pmdf cnbuild can optionally produce an option file that contains correct table sizes to hold the configuration that was just compiled (plus a little room for growth). The –option_file qualifier causes this file to be output. By default, this file is the file named by the PMDF_OPTION_FILE option in the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or PMDF Tailor Registry key (NT). The value on the –option_file qualifier may be used to specify an alternate file name. If the –nooption_file qualifier is given, then no option file will be output.

pmdf cnbuild always reads any option file that is already present via the PMDF_OPTION_FILE option of the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or PMDF Tailor Registry key (NT); use of this qualifier will not alter this behavior. However, use of the –maximum qualifier causes pmdf cnbuild to read PMDF options from the file maximum.dat in the PMDF table directory in addition to reading the file named by PMDF_OPTION_FILE. This file specifies near maximum table sizes. Only use this qualifier if the current table sizes are inadequate, and only use it to create a new option file. The –noimage_file qualifier should always be specified when –maximum is specified since a maximum-size image would be truly enormous and extremely wasteful.

–remove
Remove any existant compiled configuration; *i.e.*, remove the file named by the PMDF_CONFIG_DATA option of the PMDF tailor file, /etc/pmdf_tailor, (UNIX) or PMDF Tailor Registry key (NT).

–sizes
–nosizes *(default)*
The –sizes qualifier instructs pmdf cnbuild to output information on the sizes of uncompiled PMDF tables.

–statistics
–nostatistics *(default)*
The –statistics qualifier instructs pmdf cnbuild to output information on how much of the various tables in the compiled configuration were actually used to store data. These numbers give a rough measurement of the efficiency of the compilation, and may indicate whether or not an additional rebuild with the –option_file qualifier is needed.

EXAMPLES

1
```
# pmdf cnbuild
```

This is the standard command used on UNIX to regenerate a compiled configuration. After compiling the configuration, restart any programs which may need to reload the new configuration; *e.g.*, the TCP SMTP server should be restarted.

2
```
# pmdf cnbuild -noimage_file -option_file -maximum
# pmdf cnbuild
```

Use these two UNIX commands when you encounter the infamous "No room in table" error message.

3
```
C:\> pmdf cnbuild
```

This is the standard command used on NT to regenerate a compiled configuration. After compiling the configuration, restart any programs which may need to reload the new configuration; *e.g.*, the Dispatcher should be restarted.

4
```
C:\> pmdf cnbuild -noimage_file -option_file -maximum
C:\> pmdf cnbuild
```

Use these two NT commands when you encounter the infamous "No room in table" error message.

configure—Create PMDF configuration file

Create a basic PMDF configuration file and alias file on UNIX.

SYNTAX

pmdf configure *product-or-component-name*

restrictions Must have superuser privileges in order to use this utility. This utility is not available on NT; on NT, use the web-based configuration utility.

PARAMETERS

product-or-component-name
The product name, *i.e.*, `mta`, `access`, `firewall`, `lan`, or component name, *i.e.*, `dispatcher` or `mailbox_servers`.

DESCRIPTION

`pmdf configure` is an interactive command line utility for creating basic PMDF configuration files. Note that there is also a newer, web-based configuration utility for generating PMDF-MTA (including mailbox servers) configurations; see the *PMDF Installation Guide* for additional details.

Most fundamentally, it is used to generate a basic PMDF configuration file, alias file, mappings file, and security configuration file, usually `/pmdf/table/pmdf.cnf`, `/pmdf/table/aliases`, `/pmdf/table/mappings`, and `/pmdf/table/security.cnf`, respectively. The utility prompts for answers to questions regarding a site's node names and network connectivity, and then creates the basic files in accord with the answers to those questions.

The utility is also used to configure optional PMDF layered products, and to configure various PMDF components.

This utility is usually run when PMDF is first installed. It may also be convenient to run this utility, rather than manually editing the PMDF configuration file, after changes in a site's network configuration, such as the addition or removal of nodes, or a change in the status of a site's access to a larger network. Although by default this utility writes "live" files, overwriting any existing configuration and alias file, different file names may be specified, which can be useful for comparison or testing purposes. Note that since this utility does not take into account any pre-existing configuration file and alias file, any manual changes made to such files must be re-entered into the new files.

For a complete description and examples of using this utility to create configuration files for PMDF products, see the *PMDF Installation Guide*. For a description of using this utility to create a PMDF Service Dispatcher configuration

file, see Chapter 11. For a description of using this utility to configure the PMDF POP3 and IMAP servers, see Chapter 13.

convertdb—Convert a V6.0-V6.4 PMDF crdb database to a V6.5 or later PMDF crdb database

Convert the contents of a V6.0 to V6.4 PMDF for UNIX `crdb` database (a Berkeley DB / SleepyCat database) to a V6.5 or later PMDF `crdb` database (a PBL database).

SYNTAX

pmdf convertdb *input-database-spec output-database-spec*

Command Qualifiers	Defaults
None.	*None.*

restrictions

This utility will *not* convert databases from versions of PMDF prior to V6.0. If you are upgrading from V5.2 or earlier, you must first upgrade to a version of PMDF between V6.0 and V6.4 and convert your database to Berkeley DB (SleepyCat) format, and then upgrade to V6.5 or later and convert the Berkeley DB (SleepyCat) formatted databases to PBL. Alternatively, you can rebuild your databases from sources instead of converting them.

The defragment channel database (`/pmdf/table/defragment_cache.*`) is not supported for conversion. It does not need to be converted since the `defragment` channel rebuilds it based on the messages in its queue cache directory the first time the channel runs.

PARAMETERS

input-database-spec
The name of the PMDF database from which to read entries. This must be a Berkeley DB (SleepyCat) formatted database, such as one created by PMDF versions V6.0 through V6.4. Special keywords such as PMDF_ALIAS_DATABASE, PMDF_REVERSE_DATABASE, PMDF_FORWARD_DATABASE, PMDF_GENERAL_DATABASE, PMDF_DOMAIN_DATABASE, and PMDF_PIPE_DATABASE are supported; the use of such a special keyword tells PMDF to convert the database specified by the corresponding PMDF tailor file option.

output-database-spec
The name of the V6.5 or later PMDF database to which to write the entries stored in the input PMDF database. Special keywords such as PMDF_ALIAS_DATABASE, PMDF_REVERSE_DATABASE, PMDF_FORWARD_DATABASE, PMDF_GENERAL_DATABASE, PMDF_DOMAIN_DATABASE, and PMDF_PIPE_DATABASE are supported; the use of such a special keyword tells PMDF to write the database specified by the corresponding PMDF tailor file option.

DESCRIPTION

The format of PMDF `crdb` databases has changed with PMDF version V6.5. The `convertdb` utility reads the entries in a V6.0 through V6.4 PMDF `crdb` database (which is in Berkeley DB / SleepyCat format) and writes out the entries to a current format (PMDF V6.5 or later) PMDF `crdb` database.

EXAMPLES

1 # **pmdf convertdb PMDF_ALIAS_DATABASE PMDF_ALIAS_DATABASE**

This example shows converting a PMDF for UNIX alias database to the most current format. The input database would be a PMDF alias database in Berkeley DB (SleepyCat) format, being converted to PMDF V6.5 or later format.

counters -clear

Clear the in-memory cache of channel counters.

SYNTAX

pmdf counters -clear

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX platforms, the process must have the same UID as either the `root` or `pmdf` accounts.

PARAMETERS

None.

DESCRIPTION

To zero the in-memory channel counters, issue a `pmdf counters –clear` command.

The `pmdf counters –clear` command will create a new memory section, if one does not already exist. The values in the in-memory section will be set to zero, and then the stored messages, recipients, and volumes fields will be set from the values currently in the PMDF queue cache database.

Since some initial values will be set based on entries in the PMDF queue cache database, you may want to issue the UNIX command

```
# pmdf cache –synchronize
```

or the NT command

```
C:\> pmdf cache –synchronize
```

before clearing the counters, to ensure that the queue cache database entries are current.

counters -show

Display the contents of the in-memory cache of channel counters.

SYNTAX

pmdf counters -show

Command Qualifiers	Defaults
`-associations`	`-associations`
`-channels`	`-channels`
`-headers`	`-headers`
`-output=`*file-spec*	*None*
`-today`	`-today`

restrictions On UNIX, normally none, but if a new in-memory section must be created then privileges sufficient to create such a section are required.

DESCRIPTION

The contents of the in-memory cache of channel counters may be displayed with the `pmdf counters -show` command.

A new in-memory section will be created if one does not already exist. Note that if a new in-memory section must be created, the initial values for the number of messages stored, number of recipients, and message volumes will be set based on the entries in the PMDF queue cache database.

COMMAND QUALIFIERS

`-associations` ***(default)***
`-noassociations`
This qualifier specifies whether to show the in-memory cache of association counters.

`-channels` ***(default)***
`-nochannels`
This qualifier specifies whether to show the in-memory cache of channel counters.

`-headers` ***(default)***
`-noheaders`
Controls whether or not a header line describing each column in the table of counters is output.

`-output=`*file-spec*
Direct the output to the specified file. By default the output appears on your display.

-today *(default)*
-notoday
This qualifier specifies whether to show PMDF's count for the number of messages processed this day. Note that the "today" count, like all counters counts, is a "lightweight" count; in case of any problems updating the counter, PMDF skips the update and keeps on processing. So

EXAMPLES

This UNIX example shows displaying the counters for all channels and associations.

```
# pmdf counters -show
4263 messages processed so far today
30000 messages per day are permitted by your license

Channel                      Messages  Recipients      Blocks
-----------------------  ----------  ----------  ----------
l
    Received                     3863        3881       25786
    Stored                         89          89         460
    Delivered                    3876        3894       26018 (3859 first time)
    Submitted                      99         114        1611
    Attempted                      17          17          25
    Rejected                        0           0           0
    Failed                          1           1           6

    Queue time/count        29794837/3877 = 7.68502E3
    Queue first time/count  18904343/3860 = 4.8975E3

tcp_local
    Received                     208         217        4153
    Stored                         3           3           9
    Delivered                    200         212        2461 (197 first time)
    Submitted                   4053        4078       25919
    Attempted                      7           7           0
    Rejected                      46          68           0
    Failed                        14          14        1695

    Queue time/count        1106266/211 = 5.24297E3
    Queue first time/count  455897/208 = 2.19181E3

    Current In Assocs            127
    Total In Assocs             1056
    Total Out Assocs             132
    Rejected Out Assocs           11
    Failed Out Assocs              1

Channel      Timestamp    Association
------------ ------------ ------------------------------------------------
tcp_local    01-Feb 00:27 TCP|192.160.253.70|25|192.160.253.66|3465
tcp_local    25-Jan 00:31 TCP|192.160.253.70|25|192.160.253.66|3496
tcp_local    26-Jan 14:50 TCP|192.160.253.70|25|192.160.253.66|2086
tcp_local    05-Feb 12:23 TCP|192.160.253.70|25|192.160.253.66|3593
tcp_local    01-Feb 00:34 TCP|192.160.253.70|25|192.160.253.66|3581

...
#
```

counters -today—Display number of messages processed today

Display PMDF's count of the number of messages processed so far today.

SYNTAX

pmdf counters -today

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

PARAMETERS

None.

DESCRIPTION

PMDF's count of the number of messages processed so far today may be displayed with the `pmdf counters –today` command.

Note that the "today" count, like all counters counts, is a "lightweight" count; in case of any problems updating the counter, PMDF skips the update and keeps on processing. So

EXAMPLES

```
# pmdf counters –today
4263 messages processed so far today
30000 messages per day are permitted by your license
```

This UNIX example shows displaying PMDF's count of the number of messages processed so far that particular day.

crdb—Create database

`pmdf crdb` is a utility used to create and update PMDF database files.

SYNTAX

pmdf crdb *input-file-spec output-database-spec*

Command Qualifiers	Defaults
`-append`	`-noappend`
`-count`	`-count`
`-delete`	`-nodelete`
`-dump`	*See text*
`-duplicates`	`-noduplicates`
`-exception_file=`*file-spec*	`-noexception_file`
`-huge_records`	`-huge_records`
`-long_records`	`-nolong_records`
`-quoted`	`-noquoted`
`-remove`	`-noremove`
`-statistics`	`-statistics`
`-strip_colons`	`-nostrip_colons`

restrictions

None.

prompts

Input file: *input-file-spec*
Output database: *output-database-spec*

PARAMETERS

input-file-spec
A text file containing the entries to be placed into the database. Each line of the text file must correspond to a single entry.

output-database-spec
The initial name string of the file to which to write the database; the database consists of several files named *`output-database-spec`*`.*`.

DESCRIPTION

`pmdf crdb` is a utility to create and or update PMDF database files. `pmdf crdb` simply converts a plain text file into PMDF database records and from them either creates a new database or updates the records in an existing database.

When run from the `root` account `pmdf crdb` will set the ownership of the database it creates to the `pmdf` account.

In general, each line of the input file must consist of a left hand side and a right hand side. The two sides are separated by one or more spaces or tabs. The left hand side is limited to 32 characters in a short database (the default variety), 80 characters in a long database, or 252 characters in a huge database. The right hand side is limited to 80 characters in a short database, 256 characters in a long database, or 1024 characters in a huge database. Spaces and tabs may not appear in the left hand side (but see the description of the `-quoted` qualifier below).

The format of the input file is described in the sections describing each particular PMDF database. For instance, the format of the input file for an alias database is described in Section 3.1.2; the format of the input file for the domain database (rewrite rule database) is described in Section 2.2.9; the format of the input file for the address reversal database is described in Section 3.3.2.

COMMAND QUALIFIERS

-append
-noappend ***(default)***
When the default, `-noappend`, qualifier is in effect, a new database is created, overwriting any old database of that name. Use the `-append` qualifier to instruct PMDF to instead add the new records to an existing database.

-count ***(default)***
-nocount
Controls whether or not a count is output after each group of 100 input lines are processed.

-delete
-nodelete ***(default)***
Use the `-delete` qualifier to instruct PMDF to delete the specified records from an existing database. The input file should contain one key value per line for the entries to delete. The data portion of the line is ignored. If the database was created with `-duplicate`, for multiple entries with the same key value, only the first entry is deleted.

-dump
`pmdf crdb -dump` is a synonym for `pmdf dumpdb`. It is used to dump an existing database to a flat text file – or to stdout if no output file is specified. The parameters are interpreted as the input database specification, and optionally a flat text file to which to write the output. No other qualifiers are valid when `-dump` is specified.

-duplicates
-noduplicates ***(default)***
Controls whether or not duplicate records are allowed in the output files. Currently duplicate records are of use only in the domain database (rewrite rules database) and databases associated with the directory channel.

-exception_file=*file-spec*
-noexception_file ***(default)***
`pmdf crdb` may encounter records that cannot be loaded into the database. This usually means that these records had keys (left hand sides) that were duplicates of other keys previously encountered in the input file. These exception records

can optionally be written to a separate output file for later examination; the –exceptions_file qualifier controls the writing of this file. Note that the lines in this file are not plain text; they are formatted as database entries.

–long_records
–nolong_records ***(default)***
–huge_records
–nohuge_records
These qualifiers control the size of the output records. By default left hand sides are limited to 32 characters and right hand sides are limited to 80 characters. If –long_records is specified, the limits are changed to 80 and 256, respectively. If –huge_records is specified, the limits are changed to 252 and 1024, respectively. Currently, –huge_records databases are supported only for the alias database.

–quoted
–noquoted ***(default)***
This qualifier controls the handling of quotes. Normally pmdf crdb pays no particular attention to double quotes. If –quoted is specified, pmdf crdb matches up double quotes in the process of determining the break between the left and right hand sides of each input line. Spaces and tabs are then allowed in the left hand side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of database keys. Note: The quotes are not removed unless the –remove qualifier is also specified.

–remove
–noremove ***(default)***
This qualifier controls the removal of quotes. If pmdf crdb is instructed to pay attention to quotes, the quotes are normally retained. If –remove is specified, pmdf crdb removes the outermost set of quotes from the left hand side of each input line. Spaces and tabs are then allowed in the left hand side if they are within a matching pair of quotes. This is useful for certain kinds of databases, where spaces may form a part of database keys. Note: –remove is ignored if –quoted is not in effect.

–statistics ***(default)***
–nostatistics
Controls whether or not some simple statistics are output by pmdf crdb, including the number of entries (lines) converted, the number of exceptions (usually duplicate records) detected, and the number of entries that could not be converted because they were too long to fit in the output database. –nostatistics suppresses output of this information.

–strip_colons
–nostrip_colons ***(default)***
The –strip_colons qualifier instructs pmdf crdb to strip a trailing colon from the right end of the left hand side of each line it reads from the input file. This is useful for turning alias file entries into an alias database.

EXAMPLES

1 # **pmdf crdb –long_records /pmdf/table/aliases.txt PMDF_ALIAS_DATABASE**

The above example shows UNIX commands that may be used to create an alias database with “long” record entries.

2 C:\>**pmdf crdb -long_records C:\pmdf\table\aliases.txt PMDF_ALIAS_DATABASE**

The above example shows NT commands that may be used to create an alias database with “long” record entries.

dumpdb—Dump contents of a PMDF crdb database to a file

Dump contents of a PMDF `crdb` database to a file.

SYNTAX

pmdf dumpdb *input-database-spec [output-file-spec]*

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

PARAMETERS

input-database-spec
The name of the database from which to read entries. PMDF by default will looks for a current format database of the given name; if none such exists, PMDF will look for an old format database of the given name. Special keywords such as PMDF_ALIAS_DATABASE, PMDF_REVERSE_DATABASE, PMDF_FORWARD_DATABASE, PMDF_GENERAL_DATABASE, PMDF_DOMAIN_DATABASE, and PMDF_PIPE_DATABASE (the symbolic names for PMDF databases) are supported; the use of such a special keyword tells PMDF to dump the database specified by the corresponding PMDF tailor file option.

output-file-spec
The name of the ASCII file to which to write the entries stored in the database. If no output file is specified, the output is written to stdout.

DESCRIPTION

The `dumpdb` utility writes the entries in a PMDF `crdb` database to a flat ASCII file. In particular, this utility may be used to write the contents of an old style database (PMDF V5.2 or earlier) to a file from which a new style database (PMDF V6.0 or later) may be built using the `pmdf crdb` utility.

EXAMPLES

The following commands may be used to dump the contents of an alias database to a file, and then to recreate the alias database from that file:

```
# pmdf dumpdb PMDF_ALIAS_DATABASE alias.tmp
# pmdf crdb alias.tmp PMDF_ALIAS_DATABASE
```

edit—Edit PMDF configuration files

Edit PMDF configuration files.

SYNTAX

pmdf edit *[filename [printer]]*

Command Qualifiers	Defaults
-P	
-PT	

restrictions This utility is only available on NT.

PARAMETERS

filename
A file name for which PMDF file to edit. This may be either a PMDF symbolic file name, *e.g.*, PMDF_CONFIG_FILE, PMDF_TABLE:PMDF.CNF, *etc.*, or a normal Windows file specification, *e.g.*, `C:\pmdf\table\pmdf.cnf`.

printer
A Windows printer specification, *e.g.*, `\\bettyo\monster`.

DESCRIPTION

The `pmdf edit` command launches a document editor similar to the Windows Notepad application. Unlike Notepad, however, `pmdf edit` is geared towards editing PMDF configuration files:

1. Allows multiple files to be edited simultaneously.
2. File Open and Save As operations default to the PMDF table directory.
3. Has printing features oriented towards printing configuration files (line number display, automatic line wrapping, etc.).

The `pmdf edit` application is the executable `\pmdf\bin\cedit.exe` and may be launched from the desktop by double-clicking on the executable or by double-clicking on a `.cnf` file associated with the application. (This association will not exist until the application has been run once.)

COMMAND QUALIFIERS

–P
Print the specified file on the default printer.

–PT
Print the specified file on the specified printer.

find—Find a version of a PMDF log file

Find specified version of a PMDF log file.

SYNTAX

pmdf find *file-pattern*

Command Qualifiers	Defaults
`-f=`*offset-from-first*	*None*
`-l=`*offset-from-last*	*None*

restrictions Must have read access to the requested file.

PARAMETERS

file-pattern
A file name pattern for which PMDF log file to find.

DESCRIPTION

The `pmdf find` utility may be used to find the precise file name of the specified "version" of a PMDF file. PMDF log files have a *-uniqueid* appended to the file name to allow for the creation of multiple "versions" of the log file; on UNIX, the *-uniqueid* is appended to the very end of the file name (the end of the file extension), while on NT, the *-uniqueid* is appended to the end of the name part of the file name, before the file extension. The `pmdf find` utility understands these unique ids and can find the particular file name corresponding to the requested "version" of the file.

The default, if no offset qualifier is specified, is to find the most recent "version" of the file.

COMMAND QUALIFIERS

-f=offset-from-first
This qualifier is used to specify finding the *n*th "version" of the file (starting counting from 0). For instance, to find the earliest (oldest) "version" of the file, specify `-f=0`

-l=offset-from-last
This qualifier is used to specify finding the *n*th from the last "version" of the file (starting decrementing from 0 as the most recent version). For instance, to find the most recent (newest) "version" of the file, specify `-l=0`

EXAMPLES

1 `# pmdf find /pmdf/log/tcp_local_slave.log`

This UNIX command will print out the file name of the `/pmdf/log/tcp_local_slave.log-`*`uniqueid`* file most recently created.

2 `# pmdf find /pmdf/log/tcp_bitnet_master.log -f=0`

This UNIX command will display the file name of the oldest `/pmdf/log/tcp_bitnet_master.log-`*`uniqueid`* file.

3 `C:\> pmdf find C:\pmdf\log\tcp_local_slave.log`

This NT command will print out the file name of the `C:\pmdf\log\tcp_local_slave-`*`uniqueid`*`.log` file most recently created.

4 `C:\> pmdf find \pmdf\log\tcp_bitnet_master.log -f=0`

This NT command will display the file name of the oldest `C:\pmdf\log\tcp_bitnet_master-`*`uniqueid`*`.log` file.

kill—Kill the specified component

Kill the specified component.

SYNTAX

pmdf kill *[component]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have the same process-id as the process to be killed, or be superuser. This utility is not available on NT.

PARAMETERS

component
The PMDF component to be killed. Valid values are `job_controller`, `dispatcher`. If no parameter is specified, all components are killed.

DESCRIPTION

The `pmdf kill` utility immediately and indiscriminately kills the specified process (using `kill -9`), even if that process is in the middle of transferring e-mail. So use of the `pmdf shutdown` utility, which performs an orderly shutdown, is generally preferable.

license -verify—Verify the validity of a PMDF license

On Solaris, Linux, and Windows, verify the validity of a PMDF license file.

SYNTAX

pmdf license -verify *[file-spec]*

restrictions

PARAMETERS

file-spec
The PMDF license file to check. If no file is specified, then all PMDF license files in the PMDF table directory (normally `/pmdf/table` on UNIX, or `C:\pmdf\table` on NT) are checked.

DESCRIPTION

The `pmdf license -verify` utility is available on Solaris, Linux, and Windows to check the validity of the syntax, date, and checksum in a PMDF license file. If no file is specified, then all PMDF license files in the PMDF table directory (normally `/pmdf/table` on Solaris or Linux, or `C:\pmdf\table` on Windows) are checked.

EXAMPLES

The following example shows checking a PMDF-MTA license file on a Solaris system:

```
# pmdf license -verify /pmdf/table/PMDF-MTA-SUN.license
License file check ok
```

password—Set remote authentication password

Set password for remote authentication, *e.g.*, POP client (APOP), IMAP client (CRAM), or mailbox filter authentication.

SYNTAX

pmdf password *[password]*

Command Qualifiers	Defaults
`-create`	`-create`
`-delete`	`-create`
`-service=`*keyword*	`-service=DEFAULT`
`-show`	`-create`
`-test`	`-create`
`-user=`*username*	*See text*

restrictions

On UNIX, all operations other than setting one's own password require privileges.

prompts

New password: *password*

PARAMETERS

password
The password to set. Note that APOP passwords are case sensitive.

DESCRIPTION

The `pmdf password` utility is used to create and modify PMDF password database entries. This database may be used by POP clients issuing the APOP command, by IMAP or POP clients using the CRAM-MD5 authentication mechanism, or possibly by users authenticating themselves to modify their personal mailbox filters.

Note that in general, just which source of password authentication information is used—whether the PMDF password database, or some other source—is controlled by the PMDF security configuration file; see Chapter 14. That is, a connection comes in (POP, IMAP, or mailbox filtering) and is mapped to a security rule set; the security rule set in the PMDF security configuration then controls where and how authentication is performed for that connection.

For instance, the DEFAULT security rule set in PMDF's implicit security configuration (which applies if no security configuration file exists) checks first for a PMDF popstore profile password, next for a PMDF password database entry, and finally falls through to checking for a system password entry.

Note that APOP and CRAM-MD5 passwords cannot be stored in the system password file. Therefore, in order to support use of the POP protocol's APOP command or AUTH command with CRAM-MD5, or the IMAP protocol's authenticate command with CRAM-MD5, the user must have a password entry stored in an authentication source other than (or in addition to) the system password file. The PMDF password database can be that additional authentication source.

Thus for instance, for a POP or IMAP connection handled by the DEFAULT security rule set, a user must either be a PMDF Message Store or PMDF popstore user (in which case their PMDF user profile password is normally[1] sufficient for remote authentication), or if they are a legacy UNIX message store (Berkeley mailbox) user then they must have a PMDF password database entry in addition to their system password file entry.

For mailbox filter connections handled by the DEFAULT security rule set of PMDF's implicit security configuration, authentication will be performed preferentially against the PMDF user profile, if the user has a PMDF user profile entry, if not then against the PMDF password database, if the user has an entry in it, and finally, only if the user has neither sort of entry, against the system password file.

The above discussion regards whether the PMDF password database will actually be used as the source of authentication information. When the PMDF password database is used as the source of authentication information, then an additional issue can arise, namely which of a user's possibly multiple entries will be checked for the authentication. That is, a user can have multiple entries in the PMDF password database, one for each allowed `-service` value. The sort of connection (assuming that the PMDF password database is even checked) will control which `-service` entry is preferentially checked. Note that the sort of `-service` entry checked has nothing to do with the PMDF security configuration (which instead controlled whether or not the PMDF password database was queried at all); the sort of `-service` entry checked when the PMDF password database is queried has entirely to do with which component of PMDF is doing the querying (what sort of connection this regards).

Queries by the POP server will first check a user's `-service=POP` entry, but if such an entry does not exist will fall through to the user's `-service=DEFAULT` entry. Queries by the IMAP server will first check a user's `-service=IMAP` entry, but if such an entry does not exist will fall through to the user's `-service=DEFAULT` entry.

Queries for mailbox filtering will check which channel a user matches. For a user matching a msgstore channel, the mailbox filter query will preferentially use the user's `-service=IMAP` entry, but if such an entry does not exist will fall through to the user's `-service=DEFAULT` entry. For a user matching a popstore channel, the mailbox filter query will preferentially use the user's `-service=POP` entry, but if such an entry does not exist will fall through to the user's `-service=DEFAULT` entry. For a user matching the local channel, the mailbox filter query will use the user's `-service=DEFAULT` entry.

[1] The PMDF Message Store and PMDF popstore, however, have a PWD_ELSEWHERE flag to say that passwords are stored elsewhere; if this is set, even a PMDF Message Store or PMDF popstore user might use a PMDF password database entry.

Most sites and users will not want to use –service specific password database entries. Then each user has one entry, their –service=DEFAULT entry, used whenever the PMDF password database is queried.

But for sites and users who do want to use –service specific password database entries, while the above description of –service specific probes may sound complicated, the goal is simply to query the "natural" password entry for each case.

COMMAND QUALIFIERS

–create
Create a PMDF password database entry. This qualifier is the default.

–delete
Delete a user/password entry pair from the PMDF password database.

–service=*keyword*
Specify for what service a particular password method and password value apply. The default service keyword is DEFAULT; POP3 and IMAP are other possible keywords.

–show
Show a user/service/password-method entry in the PMDF password database. Note that this commmand does not show the password value.

–test
Compare a specified password against a password stored in the PMDF password database.

–user=*username*
Set or show a password entry in the PMDF password database for the specified user. To show all users' entries specify the asterisk as a value.

EXAMPLES

To add a user jsmith with password secret to the database, use the UNIX command

```
# pmdf password –user=jsmith secret
```

The user jsmith may change his own password, with prompting so that the password is not printed on the screen, using the UNIX command

```
% pmdf password
Password:
```

To list all usernames that have an entry in the PMDF password database, use the following command:

```
# pmdf password –show –user="*"
```

ERROR MESSAGES

`cannot open password file`

The PMDF password database does not exist, or could not be opened.

`no world privilege`

Must be superuser or in the `pmdf_world` group in order to specify an entry for a user other than oneself.

process—Show currently executing PMDF jobs

List currently executing PMDF jobs.

SYNTAX

pmdf process

Command Qualifiers	Defaults
None.	*None.*

restrictions This utility is not available on NT.

PARAMETERS

None.

DESCRIPTION

Show current PMDF processes. Normally, the PMDF Service Dispatcher and PMDF Job Controller should always be present; additional processes may be present if messages are currently being processed, or if certain additional PMDF components are in use.

EXAMPLES

The following command shows current PMDF processes:

```
# pmdf process
    USER   PID S  VSZ  RSS    STIME        TIME COMMAND
    pmdf 26311 S 12264 2128   Mar_10        0:06 /pmdf/bin/dispatcher
    pmdf 26323 S 17904 1640   Mar_10        0:21 /pmdf/bin/job_controller
```

purge—Purge old log files

Purge PMDF log files.

SYNTAX

pmdf purge *[file-pattern]*

Command Qualifiers	Defaults
`-day=`*value*	*None*
`-hour=`*value*	*None*
`-num=`*value*	`-num=5`

restrictions On UNIX, must have write access to the directory containing the file(s) to be purged.

PARAMETERS

file-pattern
A file name pattern for which PMDF log files to purge. The default, if no file name pattern is specified, is to purge all the files in the PMDF log directory. Note that when specifying purging a particular sort of log file, the entire file path must be specified.

DESCRIPTION

`pmdf purge` purges back older versions of PMDF log files. (`pmdf purge` can tell which log files are older based on the `uniqueid` strings terminating PMDF log file names.)

COMMAND QUALIFIERS

`-day=`*d*
Specifying `-day=`*d* results in purging all but the last *d* days worth of log files. Note that here "day" means a 24 hour period, rather than a calendar day (midnight to midnight); *i.e.*, all but the log files created in the last 24*d* hours will be purged.

`-hour=`*h*
Specifying `-hour=`*h* results in purging all but the last *h* hours worth of log files.

`-num=`*n*
Specifying `-num=`*n* results in purging all but the last *n* log files. The default is `-num=5`.

EXAMPLES

1 `# pmdf purge`

This UNIX command will purge all but the last five versions of each sort of log file in the PMDF log directory, `/pmdf/log`.

2 `# pmdf purge -num=10 /pmdf/log/tcp_local_master.log`

This UNIX command will purge all but the last ten versions of any `/pmdf/log/tcp_local_master.log-*` files.

3 `C:\> pmdf purge`

This NT command will purge all but the last five versions of each sort of log file in the PMDF log directory, usually `C:\pmdf\log`.

4 `C:\> pmdf purge -num=10 /pmdf/log/tcp_local_master.log`

This NT command will purge all but the last ten versions of any `tcp_local_master-*.log` files in the PMDF log directory.

qclean—Hold or delete matching messages from the PMDF queue area

Hold or delete message files from the PMDF queue area that contain specified substrings in their envelope From: address, Subject: header, or message content.

SYNTAX

pmdf qclean *[channel]*

Command Qualifiers	Defaults
`-content=`*substring*	*None*
`-database`	`-database`
`-delete`	`-hold`
`-directory_tree`	`-database`
`-env_from=`*substring*	*None*
`-hold`	`-hold`
`-match=`*keyword*	`-match=AND`
`-min_length=`*n*	`-min_length=24`
`-subject=`*substring*	*None*
`-threads=`*n*	`-nothreads`
`-verbose`	`-noverbose`

restrictions

On UNIX, privileges sufficient to read and delete files in the PMDF channel queue directory tree, as well as read and update the PMDF queue cache database, are required.

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be searched for matching messages. `*` or `?` wildcard characters may be used in the channel specification.

DESCRIPTION

Hold or delete message files containing specific substrings in their envelope From: address, Subject: line, or content. By default, message files are held (`-hold`). Specify `-delete` to instead delete matching message files. The `-content`, `-env_from`, and `-subject` qualifiers are used to specify the substrings for which to search.

Any combination of `-content`, `-env_from`, and `-subject` may be specified. However, only one of each may be used. The `-match` qualifier controls whether a message file must contain all (`-match=AND`, the default) or only one of (`-match=OR`) the specified substrings in order to be held or deleted. The default is `-match=AND`.

By default, each substring to be searched for must be at least 24 bytes long (-min_length=24). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the -min_length qualifier to override this limit. Also by default, only message files identified by the queue cache database are searched (-database). Use the -directory_tree qualifier to instead search all message files actually present in the channel queue directory tree.

The optional channel parameter restricts the search to message files in the specified channel. The channel parameter may use * and ? wild cards.

The -threads qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be in the range 1-8. The default is -nothreads.

COMMAND QUALIFIERS

-content=*substring*
-env_from=*substring*
-subject=*substring*
The -content, -env_from, and -subject qualifiers are used to specify the substrings for which to search. Any combination of -content, -env_from, and -subject may be specified. However, only one of each may be used. When a combination of such qualifiers is used, the -match qualifier controls whether the qualifiers are interpreted as further restrictions (-match=AND), or as alternatives (-match=OR).

-database *(default)*
-directory_tree
The -database qualifier, the default, specifies that only message files identified by the queue cache database be searched. Use the -directory_tree qualifier to instead search all message files actually present in the channel queue directory tree.

-delete
-hold *(default)*
-hold is the default and means that matching message files will be held. Specify -delete to instead delete matching message files.

-match=*keyword*
The default is -match=AND, meaning that any criteria specified by -content, -env_from, and -subject qualifiers must all match in order for the current hold or delete operation to be applied. Specifying -match=OR means that a message will match as long as at least one such criterion matches.

-min_length=*n*
By default, each substring to be searched for must be at least 24 bytes long (-min_length=24). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the -min_length qualifier to override this limit.

-threads=*n*
-nothreads *(defaults)*
The -threads qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer in the range 1-8. The default is -nothreads.

-verbose
-noverbose *(default)*
The -verbose qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

1
```
# pmdf qclean -min_length=11 -subject="real estate"
-env_from="ownership.com"
%QM-I-QCLISTING, building a list of message files to scan
                 from the queue cache
%QM-I-SCANNING, scanning 72 message files
%QM-I-SCANNED, scanned 72 message files in 3.7500 seconds
               (19.20 messages/second)
%QM-I-HELD, held 5 message files
```

The above UNIX example shows holding all message files in the PMDF queue area that have the string “real estate” in the Subject: header and have the string “ownership.com” in the envelope From: address.

2
```
C:\> pmdf qclean -min_length=12 -subject="real estate"
-env_from="ownership.com"
%QM-I-QCLISTING, building a list of message files to scan from
                 the queue cache
%QM-I-SCANNING, scanning 72 message files
%QM-I-SCANNED, scanned 72 message files in 3.7500 seconds
               (19.20 messages/second)
%QM-I-HELD, held 5 message files
```

The above NT example shows holding all message files in the PMDF queue area that have the string “real estate” in the Subject: header and have the string “ownership.com” in the envelope From: address.

qtop—Display frequently occurring fields in PMDF queue area messages

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues.

SYNTAX

pmdf qtop *[channel]*

Command Qualifiers	Defaults
-content=*offset-specifier*	*None*
-database	-database
-directory_tree	-database
-env_from=*offset-specifier*	*None*
-min_count=*n*	-min_count=2
-subject=*offset-specifier*	-subject=(START=1,LENGTH=2147483647)
-threads=*n*	-nothreads
-top=*n*	-top=20
-verbose	-noverbose

restrictions On UNIX, privileges sufficient to read files in the PMDF channel queue directory tree, as well as read the PMDF queue cache database, are required.

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be scanned for string frequencies. * or ? wildcard characters may be used in the channel specification.

DESCRIPTION

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues. By default, only Subject: fields are shown (-subject). Use -env_from to display frequent envelope From: fields or -content to display frequent message contents. Any combination of -content, -env_from, and -subject may be specified. However, only one of each may be used.

The optional channel parameter restricts the scan to message files in the specified channel. The channel parameter may use * and ? wild cards.

By default, the top 20 most frequently occurring fields are shown (-top=20) provided that they occur 2 or more times (-min_count=2). Use the -top and -min_count qualifiers to alter this behavior. Also by default, only message files

identified by the queue cache database are scanned (-database). Use the -directory_tree qualifier to instead scan all message files actually present in the channel queue directory tree.

The -threads qualifier may be used to accelerate scanning on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous scanning threads, specify -threads=*n*. The value n must be in the range 1-8. The default is -nothreads.

The -content, -env_from, and -subject qualifiers accept the optional qualifiers start=*n* and length=*n*. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are

```
-content=(START=1,LENGTH=256),
-env_from=(START=1,LENGTH=2147483647), and
-subject=(START=1,LENGTH=2147483647).
```

Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

COMMAND QUALIFIERS

-content*[=offset-specifier]*
-env_from*[=offset-specifier]*
-subject*[=offset-specifier]*
The -content, -env_from, and -subject qualifiers are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (-subject). Use -env_from to display frequent envelope From: fields or -content to display frequent message contents. Any combination of -content, -env_from, and -subject may be specified. However, only one of each may be used.

The -content, -env_from, and -subject qualifiers accept the optional qualifiers START=*n* and LENGTH=*n*. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are

```
-content=(START=1,LENGTH=256),
-env_from=(START=1,LENGTH=2147483647), and
-subject=(START=1,LENGTH=2147483647).
```

Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

-database *(default)*
-directory_tree
The -database qualifier, the default, specifies that only message files identified by the queue cache database be searched. Use the -directory_tree qualifier to instead search all message files actually present in the channel queue directory tree.

-min_count=*n*
By default, a string must occur at least 2 times, -min_count=2, in order to be displayed.

-threads=*n*
-threads *(default)*
The -threads qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer in the range 1-8. The default is -nothreads.

-top=*n*
By default, the top 20 most frequently occurring fields are shown, (-top=20).

-verbose
-noverbose *(default)*
The -verbose qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

1
```
# pmdf qtop -subject -env_from
%QM-I-QCLISTING, building a list of message files to scan from
                 the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds
               (130.36 messages/second)
Top 20 Envelope From: addresses which occur 2 or more times
 Count  Envelope From: address
======  ================================================================
    27
    10  owner-ex-list@example.com
     2  owner-test-list@example.com

Top 20 Subject: header lines which occur 2 or more times
 Count  Subject
======  ================================================================
     6  Re: your ex-list posting
     2  Test posting to test-list
```

The above UNIX example shows displaying the most frequently occurring Subject: and envelope From: addresses amongst messages in the PMDF queue area.

2
```
C:\> pmdf qtop -subject=START=12 -min_count=15
%QM-I-QCLISTING, building a list of message files to scan from
                 the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds
               (130.36 messages/second)
Top 20 Subject: header lines which occur 15 or more times
 Count  Subject
======  ================================================================
    25  ake money fast $$$
```

The above NT example shows displaying the most frequently occuring Subject: lines that occur 20 times or more, starting from 12 characters into the Subject: header value. This may be useful when trying to spot spam that inserts random characters at the beginning of the Subject: header value.

restart—Restart PMDF jobs

Restart PMDF components.

SYNTAX

pmdf restart *[component]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

component
Optional parameter which specifies a specific PMDF component to be restarted. On UNIX, the components which may be restarted with this utility are `job_controller`, `dispatcher`, `smtp`, `pop3`, `imap`, `http`, and `poppassd`. Note that restarting the PMDF Service Dispatcher, *i.e.*, the `dispatcher` component, effectively restarts all the service components it handles, which may include `smtp`, `pop3` (the system mailbox POP server), `pop_server` (the PMDF MessageStore POP server), `imap` (the system mailbox IMAP server), `imap_server` (the PMDF MessageStore IMAP server), `http`, and `poppassd` servers. If no component name is given, then all active components will be restarted.

DESCRIPTION

Detached PMDF processes should be restarted whenever the PMDF configuration is altered—these processes load information from the configuration once only and need to be restarted in order for configuration changes to become visible to them. In addition to general PMDF configuration files such as the `pmdf.cnf` file, note that components such as the Service Dispatcher have their own specific configuration files, *e.g.*, `dispatcher.cnf`, and should be restarted after changes to any of these files.

If no component name is specified, then the `pmdf restart` utility stops any old PMDF Job Controller or PMDF Service Dispatcher jobs that might be running, and restarts the PMDF Job Controller and PMDF Service Dispatcher. If a component parameter is specified, then only detached processes associated with that component will be restarted.

On UNIX, the standard component names are:

Component	Description
circuit_check	Detached process that monitors loopback message delivery.
dispatcher	PMDF multithreaded Service Dispatcher handling services such as SMTP, POP, IMAP, and HTTP servers.
http	HTTP server processes.
imap	This restarts IMAP server processes serving out PMDF MessageStore mailboxes, and restarts IMAP server processes serving out system mailboxes.
imap_server	IMAP server processes serving out PMDF MessageStores mailboxes.
job_controller	PMDF Job Controller.
pop_server	PMDF MessageStore POP server processes (serving out PMDF MessageStore mailboxes and PMDF popstore mailboxes).
pop3	This restarts PMDF MessageStore POP server processes (serving out PMDF MessageStore mailboxes and PMDF popstore mailboxes), and also restarts POP server processes serving out system mailboxes.
poppassd	POPPASSD server processes.
smtp	SMTP server processes.

EXAMPLES

1 `# pmdf restart`

The above UNIX command restarts the PMDF jobs.

2 `C:\> pmdf restart`

The above NT command restarts the PMDF jobs.

return—Return a mail message

Return (bounce) a mail message to its originator.

SYNTAX

pmdf return *message-file-spec*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

message-file-spec
File specification of the message file to return. The specification may include wildcards, but if so, the specification must be quoted.

DESCRIPTION

The `pmdf return` utility returns a message to the message's originator. The returned message is in two parts. The first part explains the reason why the message is being returned; the text of the reason is contained in the file `return_bounced.txt` file located in the PMDF language-specific directory. The second part of the returned message contains the original message itself.

EXAMPLES

1 `# pmdf return '/pmdf/queue/l/*'`

The above UNIX command will cause all of the messages currently in the local channel, l, to be returned to their respective originators (*i.e.*, "bounced").

2 `C:\> pmdf return 'C:\pmdf\queue\tcp_local\*'`

The above NT command will cause all of the messages currently in the tcp_local channel to be returned to their respective originators (*i.e.*, "bounced").

run—Process messages in a specified channel

Process messages in a specified channel.

SYNTAX

pmdf run *channel [poll]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

channel
This parameter specifies the channel to be processed.

poll
If `poll` is not specified, the channel program will only attempt to run if there are actually messages for that channel waiting to be processed.

DESCRIPTION

The `pmdf run` utility processes the messages in the channel specified by the *`channel`* parameter. Output during processing will be displayed at your terminal, which will be tied up for the duration of the operation of the utility.

See also the `pmdf submit` and `pmdf submit_master` utilities, which, unlike `pmdf run`, will not tie up your terminal.

EXAMPLES

1 `# pmdf run tcp_local poll`

The above UNIX command may be used to process any messages in the `tcp_local` channel.

2 `C:\> pmdf run tcp_local poll`

The above NT command may be used to process any messages in the `tcp_local` channel.

shutdown—Shut down PMDF jobs

Shut down PMDF components.

SYNTAX

pmdf shutdown *[component]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

component
Optional parameter which specifies a specific PMDF component to be shut down. On UNIX, the components that may be specified are `job_controller`, `dispatcher`, `smtp`, `pop_server` (the PMDF MessageStore POP server), `pop3` (the system mailbox POP server), `imap` (the system mailbox IMAP server), `imap_server` (the PMDF MessageStore IMAP server), `http`, `poppassd`, or `circuit_check`, or indeed any specific Dispatcher service by name (as defined in the Dispatcher configuration file). Note that shutting down the PMDF Service Dispatcher, *i.e.*, the `dispatcher` component, shuts down all the service components it handles, which may include `smtp`, `pop3`, `imap`, `http`, and `poppassd`, and any other Dispatcher services. If no component name is given then all active components will be shutdown.

DESCRIPTION

The `pmdf shutdown` utility shuts down the PMDF Job Controller and the PMDF Service Dispatcher. Note that shutting down the PMDF Service Dispatcher shuts down all services (*e.g.*, SMTP, POP3, POPPASSD, IMAP, and HTTP.

Note that on NT, an alternate way of shutting down the Dispatcher and Job Controller is to go to the Services icon under the Control Panel, and then stop them.

On UNIX, `pmdf shutdown force` can be used to shut down all PMDF processes forcefully, and `pmdf shutdown wait` can be used to wait until all PMDF processes have terminated on their own. Both of these commands must be used with caution because normally server processes like IMAP will stay up as long as there are active connections.

EXAMPLES

1 `# pmdf shutdown`

The above UNIX command shuts down the PMDF jobs.

2 `C:\> pmdf shutdown`

The above NT command shuts down the PMDF jobs.

startup—Start PMDF jobs

Start the PMDF Job Controller and the PMDF Service Dispatcher.

SYNTAX

pmdf startup *[component]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

component
Optional parameter which specifies a specific PMDF component to be started: `job_controller` or `dispatcher` or `circuit_check`. If no component name is given, then all active components will be started. Alternatively, `post` may be specified to cause the PMDF periodic delivery job to be executed immediately, or `return` may be specified to cause the PMDF periodic return job to be executed immediately.

DESCRIPTION

The `pmdf startup` utility starts up detached PMDF processes. If no component parameter is specified, then the PMDF Job Controller and PMDF Service Dispatcher are started. Note that starting the Service Dispatcher starts all services the Service Dispatcher is configured to handle, which may include, for instance, SMTP, POP3, POP_SERVER, IMAP_SERVER, IMAP, and HTTP servers. If a component parameter is specified, then only detached process associated with that component will be started. The standard component names are

Component	Description
`circuit_check`	Detached loopback message monitor process.
`dispatcher`	Multithreaded Service Dispatcher.
`job_controller`	Job Controller handling PMDF processing jobs
`post`	Run the PMDF periodic delivery job
`return`	Run the PMDF periodic return job

Note the services handled by the PMDF multithreaded Service Dispatcher must be started by starting the PMDF Service Dispatcher; only services *not* being handled by the PMDF Service Dispatcher can be individually started via the `pmdf startup` utility. The Service Dispatcher may be configured to handle various services, *e.g.*, the multithreaded SMTP, POP3, POP_SERVER, IMAP, IMAP_SERVER, and HTTP servers. See Chapter 11 for details.

EXAMPLES

1 `# pmdf startup`

The above UNIX command starts the PMDF Job Controller and PMDF Service Dispatcher.

2 `C:\> pmdf startup`

The above NT command starts the PMDF Job Controller and PMDF Service Dispatcher.

submit—Process messages in a specified channel

Process messages in a specified channel.

SYNTAX

pmdf submit *channel [poll]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

channel
This parameter specifies the channel to be processed.

poll
If `poll` is not specified, the channel program will only attempt to run if there are actually messages for that channel waiting to be processed.

DESCRIPTION

The `pmdf submit` utility forks a process to process the messages in the channel specified by the *`channel`* parameter.

See also the synonymous `pmdf submit_master` utility, and the `pmdf run` utility, which runs at your terminal, rather than as a child process.

EXAMPLES

1 `# pmdf submit tcp_local poll`

The above UNIX command may be used to process any messages in the tcp_local channel.

2 `C:\> pmdf submit tcp_local poll`

The above NT command may be used to process any messages in the tcp_local channel.

submit_master—Process messages in a specified channel

Process messages in a specified channel —a synonym for `pmdf submit`.

SYNTAX

pmdf submit_master *channel [poll]*

Command Qualifiers	Defaults
None.	*None.*

restrictions On UNIX, must have superuser privileges in order to use this utility.

PARAMETERS

channel
This parameter specifies the channel to be processed.

poll
If `poll` is not specified, the channel program will only attempt to run if there are actually messages for that channel waiting to be processed.

DESCRIPTION

`pmdf submit_master` is a synonym for `pmdf submit`.

The `pmdf submit` utility forks a process to process the messages in the channel specified by the *`channel`* parameter.

See also the synonymous `pmdf submit` utility, and the `pmdf run` utility, which runs at your terminal, rather than as a child process.

EXAMPLES

1 `# pmdf submit_master tcp_local poll`

The above UNIX command may be used to process any messages in the tcp_local channel.

2 `C:\> pmdf submit_master tcp_local poll`

The above NT command may be used to process any messages in the tcp_local channel.

test -mapping—Test a mapping table

Test a mapping table in the mapping file.

SYNTAX

pmdf test -mapping *[input-string]*

Command Qualifiers	Defaults
`-flags=`*list-of-characters*	`-noflags`
`-image_file`	`-image_file`
`-mapping_file=`*file-spec*	`-mapping_file=PMDF_MAPPING_FILE`
`-option_file=`*file-spec*	`-option_file=PMDF_OPTION_FILE`
`-table=`*table-name*	*None*

restrictions

None.

prompts

Enter table name: *table-name*

PARAMETERS

input-string
Optional input string to map.

DESCRIPTION

`pmdf test -mapping` may be used to test the behavior of a mapping table in the mapping file. The string resulting from mapping an input string will be output along with a list of any metacharacters specified in the output string.

If an input string is supplied on the command line, then only the result of mapping that input string will be output. If no input string is specified `pmdf test -mapping` will enter a loop, prompting for an input string, mapping that string, and prompting again for another input string. `pmdf test -mapping` will exit when a CTRL/D (UNIX) or CTRL/Z (NT) is entered.

COMMAND QUALIFIERS

`-flags=`*list-of-characters*
`-noflags`
The `-flags` qualifier is used to specify particular flags to set during the mapping testing; for instance, the E (envelope), B (header/body), or I (message id) flags when testing a REVERSE mapping.

-image_file
-noimage_file
When the **-image_file** qualifier is specified (the default), PMDF will load the compiled configuration file PMDF_CONFIG_DATA. When -noimage_file is specified, pmdf test -mapping will unconditionally ignore any compiled mapping information and instead read mapping information from the mapping file itself.

-mapping_file=*filename*
This qualifier instructs pmdf test -mapping to use the specified mapping file rather than the default mapping file, PMDF_MAPPING_FILE.

This qualifier has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of mapping file.

-option_file=*filename*
-nooption_file
This qualifier instructs pmdf test -mapping to use the specified option file rather than the default option file PMDF_OPTION_FILE.

This qualifier has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of option file.

Use of the qualifier -nooption_file will prevent the file PMDF_OPTION_FILE from being read in when there is no compiled configuration.

-table=*table-name*
This qualifier specifies the name of the mapping table to test. If this qualifier is not specified, then pmdf test -mapping will prompt for the name of a table to use.

EXAMPLES

```
% pmdf test -mapping -noimage_file -mapping_file=/pmdf/table/pager_table.sample
Enter table name: PAGER
Input string: H|From: "Daniel C. Newman"
<dan@example.com> (Doof City)
Output string: H|F:dan
Output flags: [0, 1, 2, 'Y' 89]
Input string: ^D
%
```

In the above UNIX example, the sample PAGER mapping is tested. The -mapping_file qualifier is used to select the mapping file pager_table.sample instead of the default mapping file.

```
C:\> pmdf test -mapping -noimage_file
-mapping_file=C:\pmdf\table\mac_mappings.sample
Enter table name: MAC-TO-MIME-CONTENT-TYPES
Input string: BINHEX|7344424e|4d535744|Test.doc
Output string: APPLICATION/MSWORD
Output flags: [0, 'Y' (89)]
Input string: ^Z
C:\>
```

In the above NT example, the sample MAC-TO-MIME-CONTENT-TYPES mapping is tested. The `-mapping_file` qualifier is used to select the mapping file `mac_mappings.sample` instead of the default mapping file.

test -match—Test a mapping wildcard pattern

Test a mapping wildcard pattern.

SYNTAX

pmdf test -match

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

prompts

Pattern:	*mapping-pattern*
Target:	*target-string*

PARAMETERS

None.

DESCRIPTION

`pmdf test –match` may be used to test a mapping pattern, particularly, to test wildcard and glob matching.

When invoked, `pmdf test –match` prompts for a pattern and then for a target string to compare against the pattern, and will output whether or not the target string matched and if it did match, which characters in the target string matched which wildcard or glob of the pattern. `pmdf test –match` will loop, prompting for input, until exitted with a CTRL/D (UNIX) or CTRL/Z (NT).

EXAMPLES

```
% pmdf test -match
Pattern: $[ax1]*@*.example.com
  [  1S] cglob [1ax]
  [  2] "@"
  [  3S] glob, req 109, reps 2
  [  4] "."
  [  5] "a"
  [  6] "c"
  [  7] "m"
  [  8] "e"
  [  9] "."
  [ 10] "c"
  [ 11] "o"
  [ 12] "m"
Target: xx11a@sys1.example.com
Match.
0 - xx11a
1 - sys1
Pattern: $[ax1]*@*.example.com
  [  1S] cglob [1ax]
  [  2] "@"
  [  3S] glob, req 109, reps 2
  [  4] "."
  [  5] "a"
  [  6] "c"
  [  7] "m"
  [  8] "e"
  [  9] "."
  [ 10] "c"
  [ 11] "o"
  [ 12] "m"
Target: 12a@node.example.com
No match.
Pattern: $[ax1]*@*.example.com
  [  1S] cglob [1ax]
  [  2] "@"
  [  3S] glob, req 109, reps 2
  [  4] "."
  [  5] "a"
  [  6] "c"
  [  7] "m"
  [  8] "e"
  [  9] "."
  [ 10] "c"
  [ 11] "o"
  [ 12] "m"
Target: 1xa@node.example.com
Match.
0 - 1xa
1 - node
Pattern: ^D
%
```

In the above UNIX example, the sample mapping pattern `$[ax1]*@*.example.com` is tested for several sample target strings.

```
C:\> pmdf test -match
Pattern: $(1.2.3.0/24)
  [  1S] ipv4 [1.2.3.0/255.255.255.0]
Target: 1.2.3.4
Match.
0 - 1.2.3.4
Pattern: $(1.2.3.0/24)
  [  1S] ipv4 [1.2.3.0/255.255.255.0]
Target: 1.2.8.0
No match.
Pattern: ^Z
C:\>
```

In the above NT example, the sample mapping pattern `$(1.2.3.0/24)` is tested or two sample target strings.

test -rewrite—Test address rewriting

Test address rewriting specified by a PMDF configuration.

SYNTAX

pmdf test -rewrite *[test-address]*

Command Qualifiers	Defaults
-alias_file=*file-spec*	-alias_file=PMDF_ALIAS_FILE
-channel	-channel
-check_expansions	-nocheck_expansions
-configuration_file=*file-spec*	-configuration_file=PMDF_CONFIG_FILE
-database=*database-list*	*See text*
-debug	-nodebug
-delivery_receipt	*See text*
-destination_channel=*channel*	*None*
-filter	-nofilter
-from=*address*	-from=postmaster@*localhost*
-grey=*setting*	-grey=0
-image_file	-image_file
-input=*input-file-spec*	-input=stdin
-local_alias=*value*	-nolocal_alias
-mapping_file=*file-spec*	-mapping_file=PMDF_MAPPING_FILE
-option_file=*file-spec*	-option_file=PMDF_OPTION_FILE
-output=*output-file-spec*	output=stdout
-read_receipt	*See text*
-reprocessing	-reprocessing
-restricted=*setting*	-restricted=0
-source_channel=*channel*	-source_channel=l

restrictions *None.*

prompts Address: *test-address*

PARAMETERS

test-address
Optional parameter specifying one or more addresses to rewrite.

DESCRIPTION

`pmdf test -rewrite` provides a straightforward test facility for examining PMDF's address rewriting and channel mapping process without actually sending any message. Various qualifiers can be used to control whether `pmdf test`

-rewrite uses the configuration text files or the compiled configuration (if present), the amount of output produced, and so on.

If a test address is specified on the command line, pmdf test -rewrite applies PMDF address rewriting to that address, reports the results, and exits. If no test address is specified, pmdf test -rewrite will enter a loop, prompting for an address, rewriting it, and prompting again for another address. pmdf test -rewrite will exit when CTRL/D (UNIX) or CTRL/Z (NT) is entered.

When testing an alias corresponding to a mailing list which has an AUTH_ or CANT_ type of named parameter controlling who may post to the list, or when testing rewriting when SEND_ACCESS or related mapping tables are in effect, note that by default pmdf test -rewrite uses as the posting address the return address of the local postmaster as specified by the PMDF option RETURN_ADDRESS. To specify a different posting address for the rewriting process, use the -from qualifier.

Must be run as superuser or by a user in the pmdf_world group.

COMMAND QUALIFIERS

-alias_file=*filename*
pmdf test -rewrite normally consults the default alias file named by the PMDF_ALIAS_FILE option of the PMDF tailor file (UNIX) or PMDF Tailor Registry entry (NT) during the rewriting process. The -alias_file qualifier specifies an alternate file for pmdf test -rewrite to use.

This qualifier has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration precludes reading any sort of alias file.

-channel *(default)*
-nochannel
This qualifier controls whether the utility outputs detailed information, *e.g.*, channel flags, regarding the channel an address matches.

-check_expansions
-nocheck_expansions *(default)*
This qualifier controls checking of alias address expansion. Normally PMDF considers the expansion of an alias to have been successful if any of the addresses to which the alias expands are legal. The -check_expansions qualifier causes a much stricter policy to be applied: pmdf test -rewrite -check_expansions checks each expanded address in detail and reports a list of any addresses, expanded or otherwise, that fail to rewrite properly. For addresses that match the L channel, PMDF also performs validity checks.

-configuration_file=*filename*
pmdf test rewrite normally consults the default configuration file named by the PMDF_CONFIG_FILE option of the PMDF tailor file (UNIX) or PMDF Tailor Registry entry (NT) during the rewriting process. The -configuration_file

qualifier specifies an alternate file to use in place of the file named by PMDF_CONFIG_FILE.

This qualifier has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any sort of configuration file.

-database=*database-list*
pmdf test -rewrite normally consults the usual PMDF databases during its operation. This qualifier is used to either disable references to various databases or to redirect the database paths to nonstandard locations.

The allowed list items are alias, noalias, personal_alias, nopersonal_alias, domain, nodomain, forward, noforward, general, nogeneral, reverse, and noreverse. The list items beginning with "no" disable use of the corresponding database. The remaining items require an associated value, which is taken to be the name of that database.

-debug
-nodebug *(default)*
The address rewriting process is capable of producing additional, detailed explanations of what actions are taken and why. The -debug qualifier enables this output; it is disabled by default.

-delivery_receipt
-nodelivery_receipt
The -delivery_receipt and -nodelivery_receipt qualifiers, which explicitly set the corresponding receipt request flags, can be useful when testing the handling of receipt requests when rewriting forwarded addresses or mailing lists.

-filter
-nofilter *(default)*
The -filter qualifier may be used to have pmdf test -rewrite output any filters (personal mailbox, channel, or system) applying for the address in question.

-destination_channel= *channel*
The -destination_channel qualifier controls for which destination or target channel pmdf test -rewrite rewrites addresses. Some address rewriting is destination channel specific; this qualifier allows control of the assumed destination channel.

-from=*address*
-nofrom
The -from qualifier controls what envelope From: address is used for access control probes and mailing list access probes. If this qualifier is omitted, the postmaster return address is used for such probes. Specifying -nofrom tells PMDF to use an empty envelope From: address for access probes.

-grey=*setting*
This qualifier controls the setting of the Grey Book flag. By default, this flag has value 0. When set to 1, -grey=1, the Grey Book flag will be set on and addresses will be rewritten using the Grey Book format.

This flag is used to force rewriting of address in accordance with the JANET (Grey Book) specifications. The most significant effect is that domain specifications

appear in reverse order, *e.g.,* edu.claremont.ymir and not ymir.claremont.edu. See Section 2.3.4.83 for further details.

Grey Book address formats are not currently used in PMDF, so this qualifier's usefulness is problematic at best.

-image_file *(default)*
-noimage_file
When the -image_file qualifier is specified (the default), pmdf test -rewrite will load the compiled configuration from the file named by the PMDF_CONFIG_DATA option in the PMDF tailor file (UNIX) or PMDF Tailor Registry entry (NT). PMDF_CONFIG_DATA is usually /pmdf/lib/config_data on UNIX, and usually C:\pmdf\lib\config_data on NT. When -noimage_file is specified, pmdf test -rewrite unconditionally ignores any previously compiled configuration and instead reads configuration information from the various text files.

-input=*input-file-spec*
By default, pmdf test -rewrite takes input from stdin. The -input qualifier may be used to specify a different source for input.

-local_alias=*value*
-nolocal_alias *(default)*
This qualifier controls the setting of an alias for the local host. PMDF supports multiple "identities" for the local host; the local host may have a different identity on each channel. This qualifier may be used to set the local host alias to the specified value; appearances of the local host in rewritten addresses will be replaced by this value.

-mapping_file*[=file-spec]*
-nomapping_file
This qualifier instructs pmdf test -rewrite to use the specified mapping file rather than the default mapping file named by the PMDF_MAPPING_FILE option in the PMDF tailor file (UNIX) or PMDF Tailor Registry entry (NT). PMDF_MAPPING_FILE usually points to the file /pmdf/table/mappings (UNIX) or C:\pmdf\table\mappings (NT).

This qualifier has no effect unless -noimage_file was specified or no compiled configuration exists; use of any compiled configuration will preclude reading the mappings file.

Use of the -nomapping_file qualifier will prevent the PMDF_MAPPING_FILE file from being read in when there is no compiled configuration.

-option_file*[=filename]*
-nooption_file
This qualifier instructs pmdf test -rewrite to use the specified option file rather than the default option file named by the PMDF_OPTION_FILE option in the PMDF tailor file (UNIX) or PMDF Tailor Registry entry (NT). PMDF_OPTION_FILE usually points to the file /pmdf/table/options (UNIX) or C:\pmdf\table\options (NT).

This qualifier has no effect unless -noimage_file is specified or no compiled configuration exists; use of any compiled configuration will preclude reading any

sort of option file.

Use of the -nooption_file qualifier will prevent the PMDF_OPTION_FILE file from being read in when there is no compiled configuration.

-output=*output_file_spec*
By default, pmdf test -rewrite writes output to stdout. The -output qualifier may be used to direct the output of pmdf test -rewrite elsewhere.

-read_receipt
-noread_receipt
The -read_receipt and -noread_receipt qualifiers, which explicitly set the corresponding receipt request flags, can be useful when testing the handling of receipt requests when rewriting forwarded addresses or mailing lists.

-reprocessing *(default)*
-noreprocessing
This flag allows the utility to display the contents of a mailing list which uses the [REPROCESS] named parameter in its alias definition.

-restricted=*setting*
This qualifier controls the setting of the restricted flag. By default, this flag has value 0. When set to 1, -restricted=1, the restricted flag will be set on and addresses will be rewritten using the restricted mailbox encoding format recommended by RFC 1137.

This flag is used to force rewriting of address mailbox names in accordance with the RFC 1137 specifications; see Section 2.3.4.57 for further details.

-source_channel=*channel*
The -source_channel qualifier controls for which source channel to rewrite addresses. Some address rewriting is source channel specific; pmdf test -rewrite normally pretends that the channel source for which it is rewriting is the local channel, l on UNIX or usually msgstore on NT.

EXAMPLES

This UNIX example shows typical output generated by pmdf test -rewrite Perhaps the single most important piece of information generated by pmdf test -rewrite is displayed on the last few lines of the output, ❻, which shows the channel to which pmdf test -rewrite would submit a message with the specified test address and the form in which the test address would be rewritten for that channel. This output is invaluable when debugging configuration problems.

```
% pmdf test -rewrite DAN@EXAMPLE.COM
❶  channel                = tcp_local
   channel description    =
   channel filter         =
❷  channel flags #0       = BIDIRECTIONAL SINGLE_SYSTEM IMMNORMAL NOSERVICEALL
   channel flags #1       = SMTP_LF MX
   channel flags #2       = POSTHEADBODY HEADERINC NOEXPROUTE
   channel flags #3       = LOGGING NOGREY RESTRICTED
   channel flags #4       = EIGHTNEGOTIATE NOHEADERTRIM NOHEADERREAD RULES
   channel flags #5       =
   channel flags #6       = LOCALUSER REPORTHEADER
   channel flags #7       = ALLOWSWITCHCHANNEL NOREMOTEHOST DATEFOUR DAYOFWEEK
   channel flags #8       = NODEFRAGMENT EXQUOTA REVERSE NOCONVERT_OCTET_STREAM
   channel flags #9       = NOTHURMAN INTERPRETENCODING INCLUDEFINAL RECEIVEDFROM
   linelength             = 998
   ddrsperfile           = 127
   channel envelope address type =  SOURCEROUTE
   channel header address type =  SOURCEROUTE
❸  channel official host  = TCP-DAEMON
   channel local alias    =
   channel queue name     =
   channel daemon name    =
   channel user name      =
   urgentnotices          =
   normalnotices          =
   nonurgentnotices       =
❹  channel group ids =
❺  backward channel       = tcp_local
   header To: address     = DAN@EXAMPLE.COM
   header From: address   = DAN@EXAMPLE.COM
   envelope To: address   = DAN@EXAMPLE.COM  (route (TCP-DAEMON,TCP-DAEMON))
   envelope From: address = DAN@EXAMPLE.COM
   name                   =
   mbox                   = DAN
Extracted address action list:
    DAN@EXAMPLE.COM
Extracted 733 address action list:
    DAN@EXAMPLE.COM
Address list expansion:
0 expansion total.
Expanded address:
   DAN@EXAMPLE.COM
❻ Submitted address list:
   tcp_local
     DAN@EXAMPLE.COM (EXAMPLE.COM) *NOTIFY FAILURES* *NOTIFY DELAYS*

❼ Submitted notifications list:
```

❶ The channel to which, after rewriting as an envelope `To:` address, the address is mapped.

❷ The flags set for the channel indicated in ❶. These flags are controlled by the channel keywords on the first line of the channel control block for the specified channel. Any unknown keywords—keywords which may have been mistyped—will be interpreted as group ids and will appear on the line ❹.

❸ The channel's official host name as specified on the second line of the channel control block for the channel indicated in ❶.

❹ Any items appearing on the first line of the channel block which were not channel keywords are interpreted as group ids. Any group ids so specified for the channel are listed on this line.

❺ The channel which the address would match if rewritten as an envelope From: address.

❻ The channel to which a message with the address DAN@EXAMPLE.COM would be queued and the envelope To: address which would be used. Here, the message would be submitted to the TCP/IP channel, `tcp_local`, using the address DAN@EXAMPLE.COM. Other information appearing here might include an explicit Errors-to: address, which, if present, appears enclosed in square brackets; or notations such as `*RR*` or `*NRR*`, indicating whether or not the message is flagged for read receipts, or notations such as `*NOTIFY FAILURES*`, `*NOTIFY DELAYS*`, `*NOTIFY SUCCESSES*`, *etc.*, indicating the message's delivery receipt mechanism and flagging.

❼ Notification addresses (reserved for future use).

ERROR MESSAGES

Usually errors reported by `pmdf test -rewrite` are not actually errors regarding `pmdf test -rewrite` in particular, but rather are the utility warning of an underlying configuration problem. For instance, "Error in mm_init: ..." sorts of errors; see Section 34.3 for a discussion of many such general error messages.

`Insufficient privileges to run test -rewrite`

Must be run as superuser or by a user in the `pmdf_world` group.

`Address list error -- unknown host or domain:`

The domain name in the specified address did not rewrite to any PMDF channel. Check that the domain name was correctly spelled. If the domain name was correct, then most likely you need a new (or changed) rewrite rule in the PMDF configuration file to handle that domain name; see Section 2.2.

`Unknown group identifier ... found on channel ...`

You do not have the specified group identifier. Check that it is truly intended to be present as a group identifier, rather than simply being a misspelled channel keyword.

test -url—Test LDAP query URL

Test an LDAP query URL.

SYNTAX

pmdf test -url *[ldap-url]*

Command Qualifiers	Defaults
`-debug`	`-nodebug`

restrictions *None.*

prompts _URL: *ldap-url*

PARAMETERS

ldap-url
LDAP URL to try resolving.

DESCRIPTION

Test an LDAP query URL.

Note that the LDAP server to query is controlled by the setting of the PMDF options LDAP_HOST and LDAP_PORT in the PMDF option file; see Section 7.3.2.

COMMAND QUALIFIERS

`-debug`
`-nodebug` ***(default)***
The testing process is capable of producing additional debug output. The `-debug` qualifier enables this output; it is disabled by default.

EXAMPLES

This example shows a sample query for a site example.com that has set the LDAP_HOST and LDAP_PORT options in the PMDF option file to point to an LDAP server containing e-mail addresses in a `mail` attribute.

```
% pmdf test -url "'ldap:///dc=example,dc=com?mail?sub?sn=doe'"
URL> Jane.Doe@example.com
URL> John.Doe@example.com
```

version—Print PMDF version number

Print PMDF version number.

SYNTAX

pmdf version

Command Qualifiers	Defaults
None.	*None.*

restrictions *None.*

PARAMETERS

None.

DESCRIPTION

`pmdf version` prints out the PMDF version number, and displays the system's name, operating system release number and version, and hardware type.

EXAMPLES

```
% pmdf version
PMDF version is PMDF V6.6
libpmdf.so version V6.6; linked 10:43:51, Feb  4 2012
SunOS nodea.example.com 5.10 Generic_105181-39 sun4u sparc SUNW,Ultra-60
```

The above UNIX example shows PMDF version information for a Solaris system running PMDF V6.6.

```
C:\> pmdf version
PMDF version is PMDF V6.6
libpmdf.dll version V6.6; linked 18:17:32, Feb  4 2012
Microsoft Windows 2003 Professional version 5.2 Service Pack 1 (Build 195)
```

The above Windows 2000 example shows PMDF version information for a Windows 2000 system running PMDF V6.6.

view—Display the specified version of a PMDF log file

Display the contents of the specified "version" of a PMDF log file.

SYNTAX

pmdf view *file-pattern*

Command Qualifiers	Defaults
`-f=`*offset-from-first*	*None*
`-l=`*offset-from-last*	*None*

restrictions On UNIX, must have read access to the requested file.

PARAMETERS

file-pattern
A file name pattern for which PMDF log file to display.

DESCRIPTION

The `pmdf view` utility may be used to display a specified "version" of a PMDF log file. PMDF log files have a *-uniqueid* appended to the file name to allow for the creation of multiple "versions" of the log file; on UNIX, the *-uniqueid* is appended to the very end of the file name (the end of the file extension), while on NT, the *-uniqueid* is appended to the end of the name part of the file name, before the file extension. The `pmdf view` utility understands these unique ids and can display the contents of the particular file corresponding to the requested "version" of the file.

The default, if no offset qualifier is specified, is to display the most recent "version" of the file.

COMMAND QUALIFIERS

-f=offset-from-first
This qualifier is used to specify displaying the *n*th "version" of the file (starting counting from 0). For instance, to display the earliest (oldest) "version" of the file, specify `-f=0`

-l=offset-from-last
This qualifier is used to specify displaying the *n*th from the last "version" of the file (starting decrementing from 0 as the most recent version). For instance, to display the most recent (newest) "version" of the file, specify `-l=0`

30.2 Interactive Utilities

PMDF has the interactive command line `pmdf qm` utility, described in Section 30.2.2, The `pmdf popstore`, `pmdf movein`, and `pmdf msgstore` utilities are described in the *PMDF popstore & MessageStore Manager's Guide*.

On UNIX, PMDF also has the interactive `pmdf profile` utility, described in Section 30.2.1, the command line `pmdf configure` utility, further details on whose use may be found in the appropriate edition of the *PMDF Installation Guide*, and the `pmdf db` utility, described in the *PMDF User's Guide, UNIX Edition*.

30.2.1 PMDF Profile: Delivery Method Utility (UNIX)

The `pmdf profile` utility sets PMDF user profile database entries. It may be used by the PMDF system manager to define delivery methods for local users, and to set the delivery method for a specific user or to set a default delivery method for local users, and may also be used by users to select among the defined delivery methods.

To create a PMDF profile database, issue a command such as

```
# su pmdf -c "pmdf crdb /dev/null PMDF_USER_PROFILE_DATABASE"
```

Then to run the `pmdf profile` utility, issue the command

```
# pmdf profile
```

Use the `exit` or `quit` command to exit `pmdf profile`. The commands accepted by this utility are summarized in Table 30–2 and described in detail subsequently.

Table 30–2 Summary of `pmdf profile` commands

`set method`	Define a delivery method
`show method`	Show the definition of a delivery method
`delete method`	Delete a delivery method definition
`set delivery`	Select a delivery method
`show delivery`	Show what delivery method is currently selected
`delete delivery`	Clear a delivery method selection

delete delivery

Clear a delivery method selection.

SYNTAX

delete delivery

Command Qualifiers	Defaults
`-default`	*See text*
`-user=`*username*	*See text*

restrictions Must be superuser in order to delete another user's choice of delivery method, or to delete the default delivery method selection.

PARAMETERS

None.

DESCRIPTION

This utility is used to delete (clear) a previous delivery method selection from the PMDF user profile database. A user may only delete their own delivery method choice. Superuser can delete the delivery method for another user with the `-user` qualifier, or may delete the default delivery method, used for all users who have not specified an explicit delivery method choice, with the `-default` qualifier.

COMMAND QUALIFIERS

`-default`
Clear the default selection of delivery method.

`-user=`*username*
Clear the delivery method selection for user `username`.

EXAMPLES

Below is an example of deleting a delivery method.

```
# pmdf profile
profile> show delivery -default
  The default delivery selection is BSD
profile> delete delivery -default
profile> exit
```

delete method

Delete a delivery method definition.

SYNTAX

delete method *method-name*

Command Qualifiers	Defaults
None.	*None.*

restrictions Must be superuser in order to delete a delivery method definition.

PARAMETERS

method-name
The name of a delivery method previously defined via the `pmdf profile set method` command.

DESCRIPTION

This utility is used to delete a delivery method definition from the PMDF user profile database. (Users who had previously selected this delivery method with the `pmdf profile set delivery` command will get the default delivery method until they select another currently defined delivery method.)

EXAMPLES

Below is an example of deleting a delivery method.

```
# pmdf profile
profile> show method -all
Delivery method(s) defined:
    BSD         /var/spool/mail/%s
    DMW         |/usr/bin/inetgrecv %s
    MIME        +/var/spool/mail/%s
profile> delete method MIME
profile> exit
```

set delivery

Select a delivery method.

SYNTAX

set delivery *method*

Command Qualifiers	Defaults
`-default`	*See text*
`-user=`*username*	*See text*

restrictions Must be superuser in order to define a delivery method for a user other than oneself, or to set a default delivery method.

PARAMETERS

method
A delivery method previously defined via the `pmdf profile set method` command.

DESCRIPTION

This utility is used to select a delivery method in the PMDF user profile database. A user may only select a delivery method for themself. Superuser can select a delivery method for another user with the `-user` qualifier, or may select a default delivery method for all users who have not specified a delivery method choice with the `-default` qualifier.

EXAMPLES

Below is an example of setting the default delivery method to DMW, and the delivery method for the PMDF system manager to BSD.

```
# pmdf profile
profile> show method -all
Method BSD is defined as: /var/spool/mail/%s
Method DMW is defined as: |/usr/bin/inetgrecv %s
Method MIME is defined as: +/var/spool/mail/%s
profile> set delivery DMW -default
profile> set delivery BSD
profile> exit
```

set method

Define a delivery method in the PMDF user profile database.

SYNTAX

set method *method-name method-command*

Command Qualifiers	Defaults
None.	*None.*

restrictions Must be superuser in order to define a delivery method.

PARAMETERS

method-name
The name to be used for the delivery method being defined. This string is not case-sensitive.

method-command
The command, in `.forward` file format, corresponding to the delivery method being defined, where the string `%s` may be used to indicate the username of the user on whose behalf the command is executed. The substitution string `%+` may be used to indicate the username plus subaddress of the user on whose behalf the command is executed. The substitution string `%d` may be used to indicate the default directory of the user on whose behalf the command is executed.

DESCRIPTION

This utility is used to define a delivery method in the PMDF user profile database. The PMDF system manager and local users will then be able to select this delivery method for delivery of their messages.

EXAMPLES

Below is an example of defining a method named Teamlinks to correspond to delivery to DEC MailWorks.

```
# pmdf profile
profile> set method Teamlinks "|/usr/bin/inetgrecv %s"
profile> show method Teamlinks
Method Teamlinks is defined as: |/usr/bin/inetgrecv %s
profile> exit
```

show delivery

Show a delivery method.

SYNTAX

show delivery

Command Qualifiers	Defaults
`-all`	*See text*
`-default`	*See text*
`-user=`*username*	*See text*

restrictions Must be superuser in order to show a delivery method for a user other than oneself, or to show the default delivery method, or to show the delivery method selected by each user.

PARAMETERS

None.

DESCRIPTION

This utility is used to show a delivery method in the PMDF user profile database. A user may only show their own selected delivery method. Superuser may show the delivery method selected by any particular user with the `-user` qualifier, may show the default delivery method for all users who have not specified a delivery method choice with the `-default` qualifier, or may show the delivery method each and every user has selected with the `-all` qualifier.

EXAMPLES

Below is an example of showing the default delivery method, and the delivery method selected for the root account.

```
# pmdf profile
profile> show delivery -default
The default delivery selection is not currently set
profile> show delivery -user=root
User root delivery selection is currently set to DMW
profile> exit
```

show method

Show a delivery method definition.

SYNTAX

show method *[method-name]*

Command Qualifiers	Defaults
`-all`	*See text*

restrictions *None.*

PARAMETERS

method-name
An optional parameter, which if specified must be the name of a previously defined delivery method.

DESCRIPTION

This command is used to show the definitions of delivery methods in the PMDF user profile database. This command may be used to show the definition of a particular delivery method, by specifying the `method` parameter, or may be used to show the definitions of all the currently defined delivery methods by instead specifying the `-all` qualifier.

COMMAND QUALIFIERS

`-all`
Show the definitions of all delivery methods.

EXAMPLES

Below is an example of showing all defined delivery methods.

```
# pmdf profile
profile> show method -all
Method BSD is defined as: /var/spool/mail/%s
Method DMW is defined as: |/usr/bin/inetgrecv %s
Method MIME is defined as: +/var/spool/mail/%s
profile> exit
```

30.2.2 qm: Queue Management Utility

pmdf qm is a utility program which allows inspection and manipulation of queued messages. pmdf qm has two modes: maintenance mode and user mode. Maintenance mode can be used to inspect and manipulate the channel queue directories and the messages contained in them. Privileges sufficient to read, create, and delete files in the channel queue directory tree as well as read and update the queue cache database are required to use maintenance mode. User mode is a very restricted version of maintenance mode which allows unprivileged users to read their own messages from the queues and to return them (bounce them) back to their originator if desired. Users' own messages are messages which they themselves have sent or were posted to a list they own. They are not messages destined for the user. User mode is documented in the *PMDF User's Guide, UNIX Edition*.

To run pmdf qm in maintenance mode, issue the UNIX command

```
# pmdf qm -maintenance
```

or the NT command

```
C:\> pmdf qm -maintenance
```

Use the exit or quit command to exit pmdf qm. The commands accepted by this utility in maintenance mode are summarized in Table 30–3 below.

Table 30–3 Summary of pmdf qm maintenance mode commands

clean	Hold or delete message files matching specified criteria
counters	Control aspects of the channel counter caches and database
date	Show current date and time
delete	Irrevocably delete the specified messages
directory	List currently queued messages
exit	Exit the utility
held	List messages which have been marked as held
help	Obtain help
history	Display message delivery history information
hold	Mark a message as held
quit	Exit the utility
read	Display message envelope and header information
release	Release held message
return	Return a message to its originator
run	Execute commands from the specified file
summarize	Display a summary listing of message files
top	Display frequently occurring strings from PMDF queue area message files
view	Control whether the channel queue directory tree or queue cache database is viewed

The command recall and editing capabilities are provided by the open source software libedit (also known as editline). By default, the standard "vi" key bindings are defined. You can change various elements of the editing environment, such as using "Emacs" key bindings instead of "vi", by creating in your home directory a file called .editrc. See the editrc manpage for more information.

clean

Hold or delete message files from the PMDF queue area that contain specified substrings in their envelope From: address, Subject: header, or message content.

SYNTAX

clean *[channel]*

Command Qualifiers	Defaults
–content=*substring*	*None*
–database	*See text*
–delete	–hold
–directory_tree	*See text*
–env_from=*substring*	*None*
–hold	–hold
–match=*keyword*	–match=AND
–min_length=*n*	–min_length=24
–subject=*substring*	*None*
–threads=*n*	–nothreads
–verbose	–noverbose

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be searched for matching messages. * or ? wildcard characters may be used in the channel specification.

DESCRIPTION

Hold or delete message files containing specific substrings in their envelope From: address, Subject: line, or content. By default, message files are held (–hold). Specify –delete to instead delete matching message files. The –content, –env_from, and –subject qualifiers are used to specify the substrings for which to search.

Any combination of –content, –env_from, and –subject may be specified. However, only one of each may be used. The –match qualifier controls whether a message file must contain all (–match=AND, the default) or only one of (–match=OR) the specified substrings in order to be held or deleted. The default is –match=AND.

By default, each substring to be searched for must be at least 24 bytes long (–min_length=24). This is a safety measure: the longer the substring, the less likely the chance of false “hits”. Use the –min_length qualifier to override this limit. The message files searched may be either all those present in the channel queue directory tree, –directory_tree, or only those files with entries

in the queue cache database, -database. Use either the view command or the -directory_tree or -database qualifier to control which files are searched.

The optional channel parameter restricts the search to message files in the specified channel. The channel parameter may use * and ? wild cards.

The -threads qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be in the range 1-8. The default is -nothreads.

COMMAND QUALIFIERS

-content=*substring*
-env_from=*substring*
-subject=*substring*
The -content, -env_from, and -subject qualifiers are used to specify the substrings for which to search. Any combination of -content, -env_from, and -subject may be specified. However, only one of each may be used. When a combination of such qualifiers is used, the -match qualifier controls whether the qualifiers are interpreted as further restrictions (-match=AND), or as alternatives (-match=OR).

-database
-directory_tree
Controls whether the message files searched are only those with entries in the queue cache database, -database, or all message files actually present in the channel queue directory tree, -directory_tree.

When neither -database nor -directory_tree is specified, then the "view" selected with the view command will be used. If no view command has been issued, then -directory_tree is assumed.

-delete
-hold ***(default)***
-hold is the default and means that matching message files will be held. Specify -delete to instead delete matching message files.

-match=*keyword*
The default is -match=AND, meaning that any criteria specified by -content, -env_from, and -subject qualifiers must all match in order for the current hold or delete operation to be applied. Specifying -match=OR means that a message will match as long as at least one such criterion matches.

-min_length=*n*
By default, each substring to be searched for must be at least 24 bytes long (-min_length=24). This is a safety measure: the longer the substring, the less likely the chance of false "hits". Use the -min_length qualifier to override this limit.

-threads=*n*
-nothreads *(defaults)*
The -threads qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer in the range 1-8. The default is -nothreads.

-verbose
-noverbose *(default)*
The -verbose qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

The following example shows holding all message files in the PMDF queue area that have the string "real estate" in the Subject: header and have the string "ownership.com" in the envelope From: address.

```
qm.maint> clean -min_length=11 -subject="real estate"
 env_from="ownership.com"
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 72 message files
%QM-I-SCANNED, scanned 72 message files in 3.7500 seconds (19.20 messages/second
)
%QM-I-HELD, held 5 message files
```

counters clear

Clear the node-specific, in-memory cache of counters.

SYNTAX

counters clear

Command Qualifiers	Defaults
–associations	–associations
–channels	–channels

PARAMETERS

None.

DESCRIPTION

To clear (zero) the counters in the in-memory cache, issue the `counters clear` command. The command creates the in-memory section of association and channel counters if it does not already exist. Then it zeros all fields in the in-memory section. Finally, the fields in the in-memory section for numbers of stored messages, message recipients, and message volumes are set based on the entries in the PMDF queue cache database.

Either the association counters, or channel counters, or both, may be cleared. The default is to clear both association and channel counters.

COMMAND QUALIFIERS

–associations *(default)*
–noassociations
This qualifier specifies whether to clear the in-memory cache of association counters.

–channels *(default)*
–nochannels
This qualifier specifies whether to clear the in-memory cache of channel counters.

counters show

Display the contents of the in-memory section of channel counters.

SYNTAX

counters show *[channel]*

Command Qualifiers	Defaults
`-header`	`-header`
`-output=`*file-spec*	*None*

PARAMETERS

channel
Optional channel name indicating the channel(s) for which to show counters. May contain wildcards.

DESCRIPTION

The contents of the in-memory channel counter section may be displayed with the `counters show` command.

Note that the output of the `pmdf qm` utility's `counters show` command is currently not as detailed as the output of the shell level `pmdf counters -show` command.

COMMAND QUALIFIERS

`-header` ***(default)***
`-noheader`
Controls whether or not a header line describing each column in the table of counters is output.

`-output=`*file-spec*
Direct the output to the specified file. By default the output appears on your display.

EXAMPLES

To display the counters information for all TCP/IP channels, use the command

```
qm.maint> counters show tcp_*
Channel                     Messages  Recipients      Blocks
------------------------  ----------  ----------  ----------
tcp_local
    Received                      33          41          95
    Stored                         0           0           0
    Delivered                     33          41          95
    Submitted                      1           1           3
tcp_internal
    Received                     632         758        1453
    Stored                         1           2          10
    Delivered                    631         756        1443
    Submitted                      3           6          12
qm.maint>
```

counters today

Display PMDF's count of the number of messages processed so far today.

SYNTAX

counters today

Command Qualifiers	Defaults
None.	*None.*

DESCRIPTION

PMDF's count of the number of messages processed so far today may be displayed with the `counters today` command.

EXAMPLES

This example illustrates displaying PMDF's count of the number of messages processed so far today.

```
qm.maint> counters today
4263 messages processed so far today
30000 messages per day are permitted by your license
qm.maint>
```

date

Show the current date and time.

SYNTAX

date

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

None.

DESCRIPTION

The `date` command may be used to show the current date and time, in RFC 822 and RFC 1123 format. It is useful for placing time stamps in log files for command scripts which periodically run `pmdf qm` to check on PMDF's channel queues.

EXAMPLES

```
qm.maint> date
Fri, 15 Nov 2012 22:52:37 -0700 (PDT)
qm.maint>
```

delete

Delete one or more messages from the channel queue directory.

SYNTAX

delete *[message-id[,...]]*

Command Qualifiers	Defaults
`-all`	`-noall`
`-channel=`*name*	*None*
`-confirm`	`-noconfirm`
`-log`	`-log`

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification number or numbers shown by a previous `directory` command. Ranges are allowed.

DESCRIPTION

The `delete` command is used to delete one or more messages from the channel queue directories. The messages to be deleted are specified by their message identification numbers shown by the most recent `directory` command. That number appears in the leftmost column of the `directory` command listing. Ambiguous message numbers must be qualified by the proper channel name with the `-channel` qualifier.

Note that the `delete` command irrevocably deletes each message it is instructed to delete: the messages are not returned to their originators nor will any further attempts to be made to deliver them to their recipients. The messages are permanently deleted. Often, it is preferable to use the `return` command so as to return the message to its originator, (*e.g.,* bounce it back to the sender).

QUALIFIERS

`-all`
`-noall` ***(default)***
Delete all messages shown by the last `directory` command. When used in conjunction with the `-channel` qualifier, only those messages shown by the last `directory` command for the specified channel will be deleted.

Unless `-noconfirm` is specified with `-all`, you will be required to confirm any `delete -all` operation.

-channel=*name*
Specifies the name of the channel from which to delete messages. Wildcards are not permitted.

-confirm
-noconfirm *(default)*
When -confirm is specified, you will be prompted to confirm each message delete operation.

-log *(default)*
-nolog
Specifies whether informational messages for each message delete operation are generated.

EXAMPLES

In the following example, the directory command is used to list the messages in the local, l, channel. Then, the delete command is used to delete messages 1, 3, 20, 21, and 22. A range specification, 20-22, is used to specify message numbers 20, 21, and 22.

```
qm.maint> directory l
Fri, 15 Nov 2012 13:43:39 -0800 (PST)
Data gathered from the queue directory tree

Channel: l                          Size Queued since
------------------------------------------------------------
    1 ZZ01HNP17LSUWY9D4DNR.00          4 15-NOV-2012 01:10:23
    2 ZZ01HNP1RP3B6G9D4DNR.00         10 15-NOV-2012 01:10:24
    3 ZZ01HNP42MAMAI9D4DNR.00          3 15-NOV-2012 01:10:24
    4 ZZ01HNP4MEWC8G9D4DNR.00          8 15-NOV-2012 06:18:57
      ...
   24 ZZ01HNP90X63ZG9D4DNR.00          6 15-NOV-2012 13:21:14
------------------------------------------------------------
Total size:                          108

24 total messages queued
qm.maint> delete 1,3,20-22
%QM-I-DELETED, deleted the message file /pmdf/queue/l/ZZ01HNP17LSUWY9D4DNR.00
%QM-I-DELETED, deleted the message file /pmdf/queue/l/ZZ01HNP42MAMAI9D4DNR.00
%QM-I-DELETED, deleted the message file /pmdf/queue/l/ZZ01HNP76RTGHY9D4DNR.00
%QM-I-DELETED, deleted the message file /pmdf/queue/l/ZZ01HNP82HTXYB9D4DNR.00
%QM-I-DELETED, deleted the message file /pmdf/queue/l/ZZ01HNP83JPOCV9D4DNR.00
qm.maint>
```

directory

List currently queued messages.

SYNTAX

directory *[channel-name]*

Command Qualifiers	Defaults
`-database`	*See text*
`-directory_tree`	*See text*
`-envelope`	`-noenvelope`
`-file_info`	`-file_info`
`-from`	*See text*
`-held`	`-noheld`
`-match`	*See text*
`-owner`	*See text*
`-to`	*See text*
`-total`	*See text*

PARAMETERS

channel-name
An optional parameter specifying the channel for which to obtain a directory listing. Wildcards are permitted.

DESCRIPTION

The `directory` command is used to show the currently queued message files in either all channel queues or a particular channel queue. In the listing, message identification numbers will appear to the left of each message file name. These numbers may be used with the `delete`, `history`, `hold`, `read`, `release`, and `return` commands so as to identify which message to operate on.

The `directory` command produces its listing by looking at either the actual queue directory tree on disk, or by looking at the queue cache database. Use either the `view` command or the `-directory_tree` or `-database` or qualifiers to control the source of information used. Note that when `-directory_tree` or `view directory_tree` or is used, the "queued since" dates are the date and time that the message file was created; when `-database` or `view database` or is used, the queued since dates are the date and time that the message was enqueued and may pre-date the actual creation date for the message file itself.

QUALIFIERS

-database
-directory_tree
Controls whether the information presented is gathered from the queue cache database, -database, or by looking at the actual directory tree containing the channel queues, -directory_tree.

When neither -database or -directory_tree or is specified, then the "view" selected with the view command will be used. If no view command has been issued, then -directory_tree is assumed.

-envelope
-noenvelope *(default)*
Use the -envelope qualifier to generate a directory listing including the envelope From: address and the list of envelope To: recipients for each listed message. By default, envelope information is not displayed as it involves opening each message file and reading through its envelope.

-file_info *(default)*
-nofile_info
By default, message file size and creation date information is gathered. However, this requires accessing each message file. Specify -nofile_info if you want to avoid that overhead.

-from=*address*
This qualifier may be used to request showing only those messages with the specified envelope From: address. This qualifier implies -envelope. To specify an empty (blank) envelope From: address, use -from=<>.

-held
-noheld *(default)*
Show information only for those channels with held messages.

-match=*keyword*
This qualifier controls the interpretation of the -from and -to qualifiers. Valid keywords are AND and OR.

-owner=*username*
This qualifier may be used to request showing only those message "owned" by the specified username. This qualifier implies -database. Note that messages submitted via SMTP with authentication (SMTP AUTH) will be considered to be owned by the username that authenticated, prefixed with the asterisk, *, character. For instance, if user jdoe submits a message from an IMAP client that successfully performs SMTP authentication, then PMDF QM will consider the owner of the message to be *jdoe, and to see such messages one would use the command

```
qm.maint> dir -owner=*jdoe
```

-to=*address*
This qualifier may be used to request showing only those messages with the specified envelope To: address. This qualifier implies -envelope.

-total
This qualifier may be used to request showing only the total number of messages, rather than listing each individual message as is the default.

EXAMPLES

1

```
qm.maint> directory tcp_*
Fri, 15 Nov 2012 14:53:39 -0800 (PST)
Data gathered from the queue directory tree

Channel: tcp_local                Size Queued since
-------------------------------------------------------------
    1 ZL01HNM78RMBP496VPJS.00        4 12-Nov-2012 09:12:29.53
    2 ZM01HNMEDX5T8E96VQDN.00       10 12-Nov-2012 12:36:41.35
    3 ZX01HNP9IO1ZAM96W55R.00        6 15-Nov-2012 13:50:06.89
    4 ZY01HNP9HTAO9696W55R.00        5 15-Nov-2012 13:49:25.61
    5 ZY01HNPBGF8JVI96W55R.00        6 15-Nov-2012 14:45:34.33
    6 ZZ01HNPBFPQ4LG96W55R.00        5 15-Nov-2012 14:45:00.01
    7 ZZ01HNPBFQ4BS896W55R.00        5 15-Nov-2012 14:45:00.53
    8 ZZ01HNPBFR5KG296W55R.00        5 15-Nov-2012 14:45:01.92
    9 ZZ01HNPBFRD2IC96W55R.00        5 15-Nov-2012 14:45:02.19
   10 ZZ01HNPBFS7VP896W55R.00        5 15-Nov-2012 14:45:03.36
   11 ZZ01HNPBFTM8YY96W55R.00        5 15-Nov-2012 14:45:05.23
   12 ZZ01HNPBFY7JYU96W55R.00        5 15-Nov-2012 14:45:11.41
   13 ZZ01HNPBGL2BYC96W55R.00        5 15-Nov-2012 14:45:42.10
-------------------------------------------------------------
Total size:                         71

Channel: tcp_gateway              Size Queued since
-------------------------------------------------------------
    1 ZY01HNP9HYJ0QK96W55R.00        6 15-Nov-2012 13:49:32.60
    2 ZY01HNP9ID452296W55R.00        6 15-Nov-2012 13:49:52.18
    3 ZZ01HNPBFT1MAC96W55R.00        5 15-Nov-2012 14:45:04.47
    4 ZZ01HNPBGH5OAM96W55R.00        5 15-Nov-2012 14:45:36.85
    5 ZZ01HNPBGZO97C96W55R.00        5 15-Nov-2012 14:46:01.73
-------------------------------------------------------------
Total size:                         27

Grand total size:                   98
28 total messages queued
qm.maint>
```

This example shows how to use the `directory` command to list the messages queued to all channels whose names match the pattern "tcp_*"; *i.e.,* all TCP/IP channels.

2

```
qm.maint> directory -held
Fri, 15 Nov 2012 13:45:18 -0800 (PST)
Data gathered from the queue directory tree

Channel: tcp_local                Size Queued since
-------------------------------------------------------------
    1 ZZG01HNM78RMBP496VPJS.HELD    10 12-NOV-2012 23:31:18.34
    2 ZZM01HNMEDX5T8E96VQDN.HELD     8  8-NOV-2012 13:36:14.89
    3 ZZX01HNP9IO1ZAM96W55R.HELD    23 29-OCT-2012 07:27:49.01
-------------------------------------------------------------
Total size:                         41

Grand total size:                   41
3 total held messages queued
qm.maint>
```

In this example, the `-held` qualifier is used to check for held messages.

exit

Exit the `pmdf qm` utility.

SYNTAX

exit

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

None.

DESCRIPTION

The `exit` and `quit` commands exit the `pmdf qm` utility.

held

List currently queued messages which have been marked as held.

SYNTAX

held *[channel-name]*

Command Qualifiers	Defaults
`-database`	*See text*
`-directory_tree`	*See text*
`-envelope`	*See text*
`-file_info`	`-file_info`
`-held`	`-held`

PARAMETERS

channel-name
An optional parameter specifying the channel for which to obtain a directory listing. Wildcards are permitted.

DESCRIPTION

The `held` command is a synonym for the `directory -held` command. See the description of the `directory` command for further information.

QUALIFIERS

`-database`
`-directory_tree`
Controls whether the information presented is gathered from the queue cache database, `-database`, or by looking at the actual directory tree containing the channel queues, `-directory_tree`.

When neither `-database` or `-directory_tree` or is specified, then the "view" selected with the `view` command will be used. If no `view` command has been issued, then `-directory_tree` is assumed.

`-envelope`
Display envelope To: and From: for the held messages listed.

`-file_info` ***(default)***
`-nofile_info`
By default, message file size and creation date information is gathered. However, this requires accessing each message file. Specify `-nofile_info` if you want to avoid that overhead.

-held ***(default)***
-noheld
Show information only for those channels with held messages.

help

Obtain help on the use of `pmdf qm`.

SYNTAX

help *[topic]*

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

topic
Optional topic to obtain help on.

DESCRIPTION

The `help` command may be used to obtain information on `pmdf qm` commands. To obtain information on all of the `pmdf qm` commands, use the command

```
qm.maint> help
```

To obtain information on individual commands or topics use the command

```
qm.maint> help topic
```

where *topic* is the name of the command or topic of interest.

history

Display message history information.

SYNTAX

history *[message-id[,...]]*

Command Qualifiers	Defaults
`-all`	`-noall`
`-channel=`*name*	*None*
`-confirm`	`-noconfirm`

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous `directory` command. Ranges are allowed.

DESCRIPTION

For many channels, delivery history information is appended to the end of each message file after an unsuccessful delivery attempt has been made. With the `history` command, this information can be displayed.

The messages to show histories for are specified by their message identification numbers shown by the most recent `directory` command. That number appears in the leftmost column of the `directory` command listing. Ambiguous message numbers must be qualified by the proper channel name with the `-channel` qualifier.

Note that history information is not recorded by some channels.

QUALIFIERS

`-all`
`-noall` ***(default)***
Display history information for all messages shown with the last `directory` command. When used in conjunction with the `-channel` qualifier, only histories of those messages shown with the last `directory` command for the specified channel will be shown.

`-channel=`*name*
Specifies the name of the channel for which to show message histories. Wild cards are not permitted.

-confirm

-noconfirm ***(default)***

When `-confirm` is specified, you will be prompted to confirm whether or not to display the history for each selected message.

hold

Mark one or more messages as being held.

SYNTAX

hold *[message-id[,...]]*

Command Qualifiers	Defaults
`-all`	`-noall`
`-channel=`*name*	*None*
`-confirm`	`-noconfirm`
`-log`	`-log`

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous `directory` command. Ranges are allowed.

DESCRIPTION

Use the `hold` command to mark as held any messages which should temporarily be placed on "hold". PMDF will not attempt to deliver any messages which are marked as held. To resume processing of a held message, use the `release` command. Messages which have been held can be listed with the `directory -held` command.

The messages to be held are specified by their message identification numbers shown by the most recent `directory` command. That number appears in the leftmost column of the `directory` command listing. Ambiguous message numbers must be qualified by the proper channel name with the `-channel` qualifier.

QUALIFIERS

`-all`
`-noall` ***(default)***
Hold all messages shown by the last `directory` command. When used in conjunction with the `-channel` qualifier, only those messages shown by the last directory command for the specified channel will be held.

Unless `-noconfirm` is specified with `-all`, you will be required to confirm any `hold -all`, operation.

`-channel=`*name*
Specifies the name of the channel from which to hold messages. Wildcards are not permitted.

–confirm
–noconfirm *(default)*
When –confirm is specified, you will be prompted to confirm each message hold operation.

–log *(default)*
–nolog
Specifies whether informational messages for each message hold operation are generated.

EXAMPLES

In the following example, the directory command is used to list the messages in the local, l, channel. Then, the hold command is used to hold messages 1, 3, 20, 21, and 22. A range specification, 20-22, is used to specify message numbers 20, 21, and 22.

```
qm.maint> directory l
Fri, 15 Nov 2012 13:43:39 -0800 (PST)
Data gathered from the queue directory tree

Channel: l                          Size Queued since
------------------------------------------------------------
    1 ZZ01HNP17LSUWY9D4DNR.00        4 15-Nov-2012 01:10:23
    2 ZZ01HNP1RP3B6G9D4DNR.00       10 15-Nov-2012 01:10:24
    3 ZZ01HNP42MAMAI9D4DNR.00        3 15-Nov-2012 01:10:24
    4 ZZ01HNP4MEWC8G9D4DNR.00        8 15-Nov-2012 06:18:57
      ...
   24 ZZ01HNP90X63ZG9D4DNR.00        6 15-Nov-2012 13:21:14
------------------------------------------------------------

24 total messages queued
qm.maint> hold 1,3,20-22
%QM-I-HELD, held the message file /pmdf/queue/l/ZZ01HNP17LSUWY9D4DNR.00
%QM-I-HELD, held the message file /pmdf/queue/l/ZZ01HNP42MAMAI9D4DNR.00
%QM-I-HELD, held the message file /pmdf/queue/l/ZZ01HNP76RTGHY9D4DNR.00
%QM-I-HELD, held the message file /pmdf/queue/l/ZZ01HNP82HTXYB9D4DNR.00
%QM-I-HELD, held the message file /pmdf/queue/l/ZZ01HNP83JPOCV9D4DNR.00
qm.maint>
```

quit

Exit the *pmdf qm* utility.

SYNTAX

quit

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

None.

DESCRIPTION

The `exit` and `quit` commands exit the `pmdf qm` utility.

read

Display message envelope and header information.

SYNTAX

read *[message-id[,...]]*

Command Qualifiers	Defaults
–all	–noall
–channel=*name*	*None*
–confirm	–noconfirm
–content	–nocontent

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous `directory` command. Ranges are allowed.

DESCRIPTION

The `read` command may be used to display envelope and header information for one or more queued messages. To also view the message content, use the `–content` qualifier.

The messages to display are specified by their message identification numbers shown by the most recent `directory` command. Those number appear in the leftmost column of the `directory` command listing. Ambiguous message numbers must be qualified by the proper channel name with the `–channel` qualifier.

QUALIFIERS

–all
–noall ***(default)***
Display all messages shown with the last `directory` command. When used in conjunction with the `–channel` qualifier, only those messages shown with the last `directory` command for the specified channel will be shown.

–channel=*name*
Specifies the name of the channel from which to display messages. Wildcards are not permitted.

–confirm
–noconfirm ***(default)***
When `–confirm` is specified, you will be prompted to confirm whether or not to display each selected message.

-content
-nocontent ***(default)***
When -content is specified, the content of the message will also be shown.

EXAMPLES

In the following example, the envelope and header information for message number 1 is displayed.

```
qm.maint> read 1
Filename: /pmdf/queue/l/ZZ01HNPFR2FUN89D4GAS.00

Message id: 1
Transport layer information:
----------------------------------------------------------------------
Envelope From: address: fresnel@example.com
Envelope To: addresses: bernoulli

Message header:
----------------------------------------------------------------------
Received: from EXAMPLE.COM by EXAMPLE.COM (PMDF V5.0-1 #8790)
 id <01HNPFR0P5OW9D4GAS@EXAMPLE.COM> for BERNOULLI@EXAMPLE.COM; Fri,
 03 Jul 2012 16:48:41 -0700 (PDT)
Date: Fri, 03 Jul 2012 16:48:40 -0700 (PDT)
From: Fresnel the tabby cat <fresnel@example.com>
To: bernoulli@example.com
Subject: catnip and catnaps
Message-id: <01HNPFR12JYA9D4GAS@EXAMPLE.COM>
MIME-version: 1.0
Content-type: TEXT/PLAIN; CHARSET=US-ASCII
Content-transfer-encoding: 7BIT

qm.maint>
```

release

Release one or more held messages.

SYNTAX

release *[message-id[,...]]*

Command Qualifiers	Defaults
`–all`	`–noall`
`–channel=`*name*	*None*
`–confirm`	`–noconfirm`
`–log`	`–log`

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous `directory –held` command. Ranges are allowed.

DESCRIPTION

Use the `release` command to release any messages previously marked as held, re-enter them in the queue cache database, and run the associated channel so the message can be processed. Messages which have been held can be listed with the `directory –held` command.

The messages to be released are specified by their message identification numbers shown by the most recent `directory` command. That number appears in the leftmost column of the `directory` command listing. Ambiguous message numbers must be qualified by the proper channel name with the `–channel` qualifier.

QUALIFIERS

`–all`
`–noall` ***(default)***
Release all messages shown by the last `directory –held` command. When used in conjunction with the `–channel` qualifier, only those messages shown by the last `directory –held` command for the specified channel will be released.

Unless `–noconfirm` is specified with `–all`, you will be required to confirm any `release –all` operation.

`–channel=`*name*
Specifies the name of the channel from which to release messages. Wildcards are not permitted.

-confirm
-noconfirm *(default)*
When -confirm is specified, you will be prompted to confirm each message release operation.

-log *(default)*
-nolog
Specifies whether informational messages for each message release operation are generated.

EXAMPLES

In the following example, the directory -held command is used to list held messages in the tcp_local channel. Then, the release command is used to release all of the held messages from that channel.

```
qm.maint> directory -held tcp_local
Fri, 15 Nov 2012 13:43:39 -0800 (PST)
Data gathered from the queue directory tree

Channel: tcp_local                Size Queued since
-------------------------------------------------------------
    1 ZZ01HNP17LSUWY9D4DNR.HELD      4 15-Nov-2012 03:12:00
    2 ZZ01HNP1RP3B6G9D4DNR.HELD     10 15-Nov-2012 11:46:23
    3 ZZ01HNP42MAMAI9D4DNR.HELD      5 15-Nov-2012 18:17:01
-------------------------------------------------------------
Total size:                         19

3 total messages queued
qm.maint> release -all
Release all message files (Y/N, default is N)? YES
%QM-I-RELEASED, released the message file
   /pmdf/queue/tcp_local/ZZ01HNP17LSUWY9D4DNR.HELD
%QM-I-RELEASED, released the message file
   /pmdf/queue/tcp_local/ZZ01HNP1RP3B6G9D4DNR.HELD
%QM-I-RELEASED, released the message file
   /pmdf/queue/tcp_local/ZZ01HNP42MAMAI9D4DNR.HELD
qm.maint>
```

return

Return a message to its sender.

SYNTAX

return *[message-id[,...]]*

Command Qualifiers	Defaults
–all	–noall
–channel=*name*	*None*
–confirm	–noconfirm
–log	–log

PARAMETERS

message-id[,...]
A comma separated list of one or more message identification numbers shown with a previous `directory` command. Ranges are allowed.

DESCRIPTION

Queued messages may be returned to their originator with the `return` command. The messages to be returned are specified by their message identification numbers shown by the most recent `directory` command. That number appears in the leftmost column of the `directory` command listing. Ambiguous message numbers must be qualified by the proper channel name with the `–channel` qualifier.

The returned message is in two parts. The first part explains the reason why the message is being returned; the text of the reason is contained in the file `return_bounced.txt` file located in the PMDF language-specific directory. The second part of the returned message contains the original message itself.

QUALIFIERS

–all
–noall ***(default)***
Return all messages shown by the last `directory` command. When used in conjunction with the `–channel` qualifier, only those messages shown by the last `directory` command for the specified channel will be returned.

Unless `–noconfirm` is specified with `–all`, you will be required to confirm any `return –all` operation.

-channel=*name*
Specifies the name of the channel from which to return messages. Wildcards are not permitted.

-confirm
-noconfirm *(default)*
When -confirm is specified, you will be prompted to confirm each message return operation.

-log *(default)*
-nolog
Specifies whether informational messages for each message return operation are generated.

run

Execute commands from a file.

SYNTAX

run *file-spec*

Command Qualifiers	Defaults
`-ignore`	`-noignore`
`-log`	`-log`

restrictions Must be able to access the file and execute the commands.

PARAMETERS

file-spec
Required parameter specifying the file to execute.

DESCRIPTION

The `run` command causes PMDF to open the specified file and read and execute each line from it. Unless `-ignore` is specified, command execution will be aborted should one of the commands generate an error. By default each command is echoed to the terminal before being executed; specify `-nolog` to suppress this echo.

QUALIFIERS

`-ignore`
`-noignore` ***(default)***
By default, command execution will be aborted should one of the commands generate an error. Specify `-ignore` if you want execution to continue even if an error occurs.

`-log` ***(default)***
`-nolog`
Specifies whether commands are echoed to the display before they are executed.

summarize

Display a summary listing of message files.

SYNTAX

summarize

Command Qualifiers	Defaults
-database	*See text*
-directory_tree	*See text*
-heading	-heading
-held	-noheld
-trailing	-trailing

PARAMETERS

None.

DESCRIPTION

Display a summary listing of message files.

COMMAND QUALIFIERS

-database
-directory_tree
Controls whether the information presented is gathered from the queue cache database, -database, or by looking at the actual directory tree containing the channel queues, -directory_tree.

When neither -database nor -directory_tree is specified, then the "view" selected with the view command will be used. If no view command has been issued, then -directory_tree is assumed.

-heading *(default)*
-noheading
Controls whether or not a heading line describing each column of output is displayed at the start of the summary listing.

-held
-noheld *(default)*
Controls whether or not to include counts of .HELD messages in the output.

-trailing *(default)*
-notrailing
Controls whether or not a trailing line with totals is displayed at the end of the summary.

EXAMPLES

1

```
qm.maint> summarize
                                                Messages
                              Channel  Queued   Size (Kb)   Oldest
--------------------------------------- -------- ---------- -----------------
                           cc_local       0        0.00
                         circuitcheck       4        7.51    8 Jun, 10:19:20
                           conversion       0        0.00
                                    l       0        0.00
                              mailserv       0        0.00
                         mime_to_x400       0        0.00
                             popstore       0        0.00
                              process       0        0.00
                            reprocess       0        0.00
                         tcp_internal      15       51.47    2 Jun, 12:10:03
                            tcp_local       0        0.00
                            wpo_local       0        0.00
                           x400_local       0        0.00
                         x400_to_mime       0        0.00
--------------------------------------- -------- ---------- -----------------
                               Totals      19       58.98

qm.maint>
```

The above UNIX example shows displaying a summary listing of message files.

2

```
qm.maint> summarize
                                                Messages
                              Channel  Queued   Size (Kb)   Oldest
--------------------------------------- -------- ---------- -----------------
                           cc_local       0        0.00
                         circuitcheck       4        7.51    8 Jun, 10:19:20
                           conversion       0        0.00
                              mailserv       0        0.00
                              msgstore       0        0.00
                              process       0        0.00
                            reprocess       0        0.00
                         tcp_internal      15       51.47    2 Jun, 12:10:03
                            tcp_local       0        0.00
                            wpo_local       0        0.00
--------------------------------------- -------- ---------- -----------------
                               Totals      19       58.98

qm.maint>
```

The above NT example shows displaying a summary listing of message files.

top

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues.

SYNTAX

top *[channel]*

Command Qualifiers	Defaults
-content=*offset-specifier*	*None*
-database	*See text*
-directory_tree	*See text*
-env_from=*offset-specifier*	*None*
-min_count=*n*	-min_count=2
-subject=*offset-specifier*	-subject=(START=1,LENGTH=2147483647)
-threads=*n*	-nothreads
-top=*n*	-top=20
-verbose	-noverbose

PARAMETERS

channel
Optional parameter which specifies a specific PMDF channel area to be scanned for string frequencies. * or ? wildcard characters may be used in the channel specification.

DESCRIPTION

Display the most frequently occurring envelope From:, Subject:, or message content fields found in message files in the channel queues. By default, only Subject: fields are shown (-subject). Use -env_from to display frequent envelope From: fields or -content to display frequent message contents. Any combination of -content, -env_from, and -subject may be specified. However, only one of each may be used.

The optional channel parameter restricts the scan to message files in the specified channel. The channel parameter may use * and ? wild cards.

By default, the top 20 most frequently occurring fields are shown (-top=20) provided that they occur 2 or more times (-min_count=2). Use the -top and -min_count qualifiers to alter this behavior. The message files scanned may be either all those present in the channel queue directory tree, or only those files with entries in the queue cache database. Use either the view command or the -directory_tree or -database qualifier to control which files are scanned.

The -threads qualifier may be used to accelerate scanning on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous scanning threads, specify -threads=*n*. The value n must be in the range 1-8. The default is -nothreads.

The -content, -env_from, and -subject qualifiers accept the optional qualifiers start=*n* and length=*n*. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are

```
-content=(START=1,LENGTH=256),
-env_from=(START=1,LENGTH=2147483647), and
-subject=(START=1,LENGTH=2147483647).
```

Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

COMMAND QUALIFIERS

-content*[=offset-specifier]*
-env_from*[=offset-specifier]*
-subject*[=offset-specifier]*
The -content, -env_from, and -subject qualifiers are used to specify which frequently occurring fields should be displayed. By default, only Subject: fields are shown (-subject). Use -env_from to display frequent envelope From: fields or -content to display frequent message contents. Any combination of -content, -env_from, and -subject may be specified. However, only one of each may be used.

The -content, -env_from, and -subject qualifiers accept the optional qualifiers START=*n* and LENGTH=*n*. These qualifiers indicate the starting offset and number of bytes in the field to consider. The defaults are

```
-content=(START=1,LENGTH=256),
-env_from=(START=1,LENGTH=2147483647), and
-subject=(START=1,LENGTH=2147483647).
```

Use of these qualifiers is useful when, for example, trying to identify occurrences of a spam message which uses random text at the start of the Subject: line.

-database
-directory_tree
Controls whether the message files scanned are only those with entries in the queue cache database, -database, or all message files actually present in the channel queue directory tree, -directory_tree.

When neither -database nor -directory_tree is specified, then the "view" selected with the view command will be used. If no view command has been issued, then -directory_tree is assumed.

-min_count=*n*
By default, a string must occur at least 2 times, -min_count=2, in order to be displayed.

-threads=*n*
-threads *(default)*
The -threads qualifier may be used to accelerate searching on multiprocessor systems by dividing the work amongst multiple, simultaneously running threads. To run *n* simultaneous searching threads, specify -threads=*n*. The value *n* must be an integer in the range 1-8. The default is -nothreads.

-top=*n*
By default, the top 20 most frequently occurring fields are shown, (-top=20).

-verbose
-noverbose *(default)*
The -verbose qualifier may be used to request that the utility print out information about what it is doing as it operates.

EXAMPLES

1
```
qm.maint> top -subject -env_from
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Envelope From: addresses which occur 2 or more times
 Count  Envelope From: address
======  ================================================================
    27
    10  owner-ex-list@example.com
     2  owner-test-list@example.com

Top 20 Subject: header lines which occur 2 or more times
 Count  Subject
======  ================================================================
     6  Re: your ex-list posting
     2  Test posting to test-list
```

The above UNIX example shows displaying the most frequently occurring Subject: and envelope From: addresses amongst messages in the PMDF queue area.

2
```
qm.maint> top -subject=START=12 -min_count=15
%QM-I-QCLISTING, building a list of message files to scan from the queue cache
%QM-I-SCANNING, scanning 73 message files
%QM-I-SCANNED, scanned 73 message files in 0.5600 seconds (130.36 messages/secon
d)
Top 20 Subject: header lines which occur 15 or more times
 Count  Subject
======  ================================================================
    25  ake money fast $$$
```

The above NT example shows displaying the most frequently occuring Subject: lines that occur 20 times or more, starting from 12 characters into the Subject: header value. This may be useful when trying to spot spam that inserts random characters at the beginning of the Subject: header value.

view

Control whether the `directory` command shows the channel queue directory tree or the queue cache database.

SYNTAX

view *type*

Command Qualifiers	Defaults
None.	*None.*

PARAMETERS

type
The type of view to use: `directory_tree` or `database`.

DESCRIPTION

The `directory` command produces its listing by looking at either the actual channel queue directory tree on disk, or by looking at the queue cache database. The `view` command controls which is used. By default, the view is the channel queue directory tree. Issue the command,

```
qm.maint> view database
qm.maint>
```

to switch to viewing the queue cache database. The command

```
qm.maint> view directory_tree
qm.maint>
```

will switch you back to viewing the channel queue directory tree. Issuing the `view` command without any parameter will restore the default behavior and is thus equivalent to the `view directory_tree` command.

31 Monitoring

For monitoring PMDF, usually the best place to start is with PMDF's optional logging of message traffic; from this basic information, sites may gather statistics such as how many messages are passing through PMDF, and answering other questions on message traffic. See Section 31.1 below which provides an overview of logging, and examples on interpreting log entries.

The web-based `QM` utility, if enabled, may be used for tasks such as scanning what messages are present in the PMDF queue area; see Section 31.2. Alternatively, the command line utility PMDF QM (OpenVMS) or `pmdf qm` (UNIX and NT) may be used to scan what messages are present in the PMDF queue area; see Section 29.2.1 or Section 30.2.2, respectively, for additional details.

The PMDF message circuit check facility may be used to monitor the health of the mail system, using loopback messages (messages that travel out from and then back to the circuit check facility) to monitor that connectivity to and from remote systems is available and to monitor the speed of delivery of such messages. See Section 31.3 which discusses configuring and using this facility.

PMDF also has facilities to collect and monitor channel counters based upon the Mail Monitoring MIB, RFC 1566. Note that counters are intended for providing real-time "snap-shots" of PMDF behavior, rather than for gathering the sort of statistics instead available from the log files. For a description of the PMDF counters, see Section 31.4 below.

PMDF provides utilities to display the counters directly; on OpenVMS see the PMDF COUNTERS and PMDF QM utilities, described in Chapter 29, or on UNIX or NT see the `pmdf counters` and `pmdf qm` utilities, described in Chapter 30.

PMDF also provides a web-based monitoring interface that can display, among other things, PMDF channel counter information; see Section 31.7 below.

On OpenVMS, sites running the HP-supplied HP Commander monitoring package (also known as Enterprise Mail Monitor, EMM, or PolyCenter MAILbus Monitor, PMM) can monitor PMDF using the PMM scanning module supplied with PMDF; see Section 31.5 below.

VMS

On OpenVMS, the Process Software TCPware TCP/IP package has an SNMPv2 subagent support; sites running this TCP/IP package can therefore serve out the PMDF counters via SNMP. See Section 31.6 below.

31.1 Logging

PMDF's optional logging of message traffic is enabled via the `logging` channel keyword, as described in Section 2.3.4.84. Enabling `logging` causes PMDF to write an entry to a `mail.log*` file each time a message passes through a PMDF channel. Such log entries can be useful if you want to get statistics on how many messages are passing through PMDF (or through particular channels), or when investigating other questions such as whether and when a message was sent or delivered.

If you are only interested in gathering statistics on the number of messages passing through a few particular PMDF channels, then you may want to enable the `logging` channel keyword on just those PMDF channels of main interest. But more generally, many sites prefer to enable logging on all PMDF channels; in particular, if you are trying to track down problems, the first step in diagnosing some problems is to notice that messages are not going to the channel you expected or intended, and having logging enabled for all channels can help you spot such issues. See Section 2.3.4.84 for details on enabling logging.

In addition to the basic information always provided when `logging` is enabled, additional, optional information fields may also be logged in the `mail.log` files, controlled via various `LOG_*` PMDF options; see Section 7.3.6. Particularly likely to be of interest are the `LOG_MESSAGE_ID`, `LOG_FILENAME`, `LOG_CONNECTION`, and `LOG_PROCESS` options. Enabling `LOG_MESSAGE_ID` allows correlation of which entries relate to which message. Enabling `LOG_FILENAME` makes it easier to immediately spot how many times delivery of a particular message file has been retried, and can be useful in understanding when PMDF does or does not split a message to multiple recipients into separate message file copies on disk. Enabling `LOG_CONNECTION` causes PMDF to log TCP/IP connections, as well as message traffic, to the `mail.log` files by default; alternatively, the `SEPARATE_CONNECTION_LOG` option may be used to specify that connection log entries instead be written to `connection.log` files. When using `LOG_CONNECTION` to cause generation of TCP/IP connection entries, additionally enabling `LOG_PROCESS` allows correlation of which connection entries correspond to which message entries.

On UNIX and NT, `mail.log` and `connection.log` entries may optionally be duplicated to syslog (UNIX) or to the event log (NT) via the `LOG_MESSAGES_SYSLOG` and `LOG_CONNECTIONS_SYSLOG` options.

The exact information of interest in the `mail.log` files tends to vary substantially from site to site. Pointers to site written freeware utilities for analyzing the PMDF log files, which you may find a useful starting point, can be found at the Process Software web site:

```
http://www.process.com
```

31.1.1 Managing the Log Files

When the `logging` keyword is enabled, all message log entries are made to the file `mail.log_current` in the PMDF log directory, (*i.e.*, `PMDF_LOG:mail.log_current` on OpenVMS or `/pmdf/log/mail.log_current` on UNIX) or, if PMDF is installed on drive C:, `C:\pmdf\log\mail.log_current` on NT). If connection logging is enabled via the LOG_CONNECTION option, connection log entries are also by default written to the `mail.log_current` file, but if SEPARATE_CONNECTION_LOG=1 has been set, then the connection log entries will instead be written to the `connection.log_current` file. The message return job, which by default runs every night around midnight, appends any existing `mail.log_yesterday` to the cumulative log file, `mail.log`, renames the current `mail.log_current` file to `mail.log_yesterday`, and then begins a new `mail.log_current` file. It also performs the analogous operations for any `connection.log*` files. Note that the setting of the RETURN_UNITS PMDF option, see Section 7.3.4, affects when the log file rollover is performed.

Note that PMDF itself never does anything to the cumulative `mail.log` file; it is up to each site to manage that log file however they choose, whether by periodically saving it to backup tape, deleting it, truncating it, or the like.

When considering how to manage the log files, note that the PMDF periodic return job will execute a site-supplied `PMDF_COM:daily_cleanup.com` (OpenVMS) or `/pmdf/bin/daily_cleanup` (UNIX) or `C:\pmdf\bin\daily_cleanup.bat` (NT) procedure, if one exists. Thus some sites may choose to supply their own `daily_cleanup` that, for instance, renames the old `mail.log` file once a week (or once a month), *etc*.

31.1.2 Log Entry Format

The format of the log file is subject to change. Currently, by default, each log file entry contains eight or nine fields,[1] *e.g.*,

```
19-Jan-2012 19:16:57.64 l           tcp_local    E 1 adam@acme.com rfc822;marlowe@examplex.com marlowe@examplex.com
          ❶          ❷          ❸          ❹ ❺ ❻          ❼                    ❽                  ❾
```

These fields are:

❶ The date and time when the entry was made, written in standard OpenVMS date and time format.

❷ The channel name for the source channel. (For the Printer channel, the first 32 characters of the print queue name are logged.)

❸ The channel name for the destination channel. (For SMTP channels when `LOG_CONNECTION` is enabled, a plus, +, indicates inbound to the SMTP server; a minus, -, indicates outbound via the SMTP client.)

❹ The type of entry; see Table 31–1.

[1] The pager and MAILSERV channels, having different information to log, use an extended logging format. See Section 26.4.3 for details on pager channel log entries. See Section 4.3.9 for details on MAILSERV channel log entries.

❺ The size of the message. This is expressed in kilobytes by default, although this default can be changed by using the BLOCK_SIZE keyword in the PMDF option file. (For the Printer channel, the size field displays the job entry number on OpenVMS, and is always 0 on UNIX. For the MAILSERV channel, the size field is always 0. For the Pager channel, the size field is the number of pages sent.)

❻ The envelope From: address. Note that for messages with an empty envelope From: address, such as notification messages, this field will be blank.

❼ The original form of the envelope To: address.

❽ The active (current) form of the envelope To: address.

❾ The delivery status (SMTP channels only).

Table 31–1 Logging Entry Codes

Entry	Description
	General
D	Successful dequeue
DA	Successful dequeue with SASL (authentication)
DS	Successful dequeue with TLS (security)
DSA	Successful dequeue with TLS and SASL (security and authentication)
E	Enqueue
EA	Successful enqueue with SASL (authentication)
ES	Successful enqueue with TLS (security)
ESA	Successful enqueue with TLS and SASL (security and authentication)
J	Rejection of attempted enqueue (rejection by slave channel program)
K	Used instead of R or W when a warning, failure, or bounce message should be generated, but the sender asked not to be notified of such events.
Q	Temporary failure to dequeue
R	Recipient address rejected on attempted dequeue (rejection by master channel program), or generation of a failure/bounce message
W	Warning message generated regarding a not-yet-delivered message
Z	Some successful recipients, but this recipient was temporarily unsuccessful; the original message file of all recipients was dequeued, and in its place a new message file for this and other unsuccessful recipients will be immediately reenqueued
	SMTP channels' LOG_CONNECTION + or - entries
C	Connection closed
O	Connection opened
X	Connection rejected
Y	Connection try failed before being established
I	ETRN command received

Table 31–1 (Cont.) Logging Entry Codes

Entry	Description
	IMAP, POP, and POPPASSD server connection entries
A	Authentication attempt failed
O	Login phase completed (either successful login or aborted connection)
C	Connection closed cleanly
X	Connection aborted (by either end)†
	MAILSERV channel
F	Error processing command; command not executed
S	Command successfully executed
	Printer channel
A	Print request failed
S	Print request succeeded
	Pager channel
A	Aborted (permanent error)
F	Temporary failure
R	Page rejected (permanent error)
S	Success
	PMDF-FAX G3_TO_FAX and FAX_TO_DATA channels
A	Delivery attempt aborted, message requeued (temporary error)
F	Delivery failed (permanent error)
M	Maximum number of delivery attempts exceeded, message not requeued (permanent error)
S	Success

†Some IMAP/POP clients close the connection without sending a LOGOUT/QUIT command, so an "X" entry can happen in normal operation with such clients.

PMDF may optionally be configured to log additional information to the log file; see the LOG_* PMDF options described in Section 7.2. With LOG_CONNECTION, LOG_FILENAME, LOG_MESSAGE_ID, LOG_NODE, LOG_NOTARY, LOG_PROCESS, and LOG_USERNAME all enabled, the format becomes as follows. (Note that the sample log entry line has been wrapped for typographic reasons; the actual log entry would appear on one physical line.)

```
19-Jan-2012 13:13:27.10 NODEA   2e2d.1 tcp_local   l          E 1 service@examplex.com rfc822;adam@acme.com
          ❶           ❿     ⓫       ❷      ❸        ❹ ❺    ❻              ❼
adam 276 PMDF_QUEUE:[l]ZZ01IWFY9ELGWM00094D.00;1 <01IWFVYLGTS499EC9Y@examplex.com> SYSTEM
❽ ⓬          ⓭                                  ❹                ⓯
examplex.com (examplex.com [192.168.253.66])
⓰           ❾
```

Where the additional fields, beyond those already discussed above, are:

⑩ The name of the node on which the channel process is running.

⑪ The process id (expressed in hexadecimal), followed by a period (dot) character and a count. If this had been a multithreaded channel entry (*e.g.*, a `tcp_*` channel entry), there would also be a thread id present between the process id and the count.

⑫ The `NOTARY` (delivery receipt request) flags for the message, expressed as an integer.

⑬ The file name in the PMDF queue area.

⑭ The message id.

⑮ The username of the executing process. Note that in the case of Dispatcher services such as the SMTP server, this will be the username of the last person to startup the Dispatcher.

⑯ This connection information consists of the sending system or channel name, such as the name presented by the sending system on the HELO/EHLO line (for incoming SMTP messages), or the enqueuing channel's official host name (for other sorts of channels). In the case of TCP/IP channels, the sending system's "real" name, that is, the symbolic name as reported by a DNS reverse lookup and/or the IP address, can also be reported within parentheses as controlled by the `ident*` channel keywords; see Section 2.3.4.40. This sample assumes use of one of these keywords, for instance us of the default `identnone` keyword, that selects display of both the name found from the DNS and IP address.

31.1.3 log_condense Utility

The `log_condense` utility scans `mail.log` format files and produces a condensed report file with one entry for each message that has passed through PMDF. This condensed log file may be used to analyze PMDF message traffic, for example by importing it into a Microsoft Excel spreadsheet or using a site-written script. Most sites will want to run `log_condense` against `mail.log_yesterday` to generate a condensed log file for the previous day's mail.

To run the `log_condense` utility:

For UNIX:

```
#  /pmdf/bin/log_condense input_log_file condensed_output_file
```

For VMS:

```
$ log_condense :== $PMDF_EXE:LOG_CONDENSE.EXE
$ log_condense input_log_file condensed_output_file
```

For Windows:

```
C:\pmdf\bin> log_condense input_log_file condensed_output_file
```

The entries generated by the `log_condense` utility have the following fields (wrapped for typographic reasons):

```
18-Jul-2012 10:16:35.45 18-Jul-2012 10:16:38.02 <01KK8H2E31XG8WW0JF@otherco>
        ❶                       ❷                       ❸

john.doe@otherco.com rfc822;jane.doe@example.com jane.doe@naples.example.com
      ❹                     ❺                              ❻

naples.example.com 4 4 E,tcp_local,tcp_internal,D 250
      ❼            ❽❾       ❿                  ⓫
```

❶ The date and time the message was accepted.

❷ The date and time the message was delivered.

❸ The unique message ID.

❹ The envelope `From:` address.

❺ The original `To:` address.

❻ The destination address after rewrites, aliases, and mappings have been applied.

❼ The host name of the system the message is being delivered to.

❽ The size of the message in blocks when it was accepted.

❾ The size of the message in blocks when it was delivered.

❿ A comma-separated list of every channel that the message passed through. The status code of the accepting channel is prepended to the list, while the status code of the delivering channel is appended to the list.

⓫ The SMTP status code returned by the system the message was delivered to.

Any field of the entry that cannot be determined from the input log file will contain a hyphen.

Note: The following options must be set in `option.dat` for `log_condense` to operate properly.

```
LOG_MESSAGEID=1
LOG_HEADER=0
LOG_FORMAT=1 or 2
SEPARATE_CONNECTION_LOG=1
```

Any log files generated without these options set as indicated cannot be analyzed by `log_condense`.

31.1.4 Examples of Message Logging

The exact field format and list of fields logged in the PMDF message logging files will vary according to exactly what logging options a site sets; see Section 2.3.4.84 for a description of the basic fields, and see the discussion of the various `LOG_*` options in Section 7.3.6 for descriptions of additional, optional fields. But there are general principles for understanding such log entries; this section will show a few examples of interpreting typical sorts of log entries.

Note: For typographic reasons, log file entries will be shown folded onto multiple lines—actual log file entries are one line per entry.

When looking over a logging file, keep in mind that on a typical system many messages are being handled at once. Typically, the entries relating to a particular message will be interspersed among entries relating to other message being processed during that same time. The basic logging information is suitable for gathering a sense of the overall numbers of messages moving through PMDF. However, if you want to correlate particular entries relating to the same message to the same recipient(s), you will probably want to enable `LOG_MESSAGE_ID`. And if you want to correlate particular messages with particular files in the PMDF queue area, or to see from the entries how many times a particular not-yet-successfully-dequeued message has had delivery reattempted, you will probably want to enable `LOG_FILENAME`. For SMTP messages (handled via a TCP/IP channel), if you want to correlate TCP connections to and from remote systems with the messages sent, you will probably want to enable `LOG_PROCESS` and some level of `LOG_CONNECTION`.

Figure 31–1 Logging: A Local User Sends an Outgoing Message

```
15-Nov-2012 19:16:57.64 l            tcp_local    E 1 ❶
 adam@example.com rfc822;marlowe@example.com marlowe@example.com ❷
15-Nov-2012 19:17:01.16 tcp_local                 D 1 ❸
 adam@example.com rfc822;marlowe@example.com marlowe@example.com ❹
 dns;thor.example.com (TCP|206.184.139.12|2788|192.160.253.66|25) ❺
 (THOR.EXAMPLE.COM -- Server ESMTP [PMDF V5.1-10 #8694]) ❻
 smtp;250 2.1.5 marlowe@example.com and options OK. ❼
```

The above entries show a basic example of the sorts of log entries one might see if a local user sends a message out an outgoing TCP/IP channel, *e.g.*, to the Internet. The lines marked with ❶ and ❷ are one entry—they would appear on one physical line in an actual log file. Similarly, the lines marked with ❸–❼ are one entry and would appear on one physical line.

❶ This line shows the date and time of an enqueue (E) from the `L` channel to the tcp_local channel of a one (1) block message.

❷ This is part of the same physical line of the log file as ❶, presented here as a separate line for typographical convenience. It shows the envelope From: address, in this case adam@example.com, and the original version and current version of the envelope `To:` address, in this case marlowe@example.com.

❸ This shows the date and time of a dequeue (D) from the `tcp_local` channel of a one (1) block message—that is, a successful send by the `tcp_local` channel to some remote SMTP server.

❹ This shows the envelope From: address, the original envelope To: address, and the current form of the envelope To: address.

❺ This shows that the actual system to which the connection was made is named thor.example.com in the DNS, that the local sending system has IP address 206.184.139.12 and is sending from port 2788, that the remote destination system has IP address 192.160.253.66 and the connection port on the remote destination system is port 25.

❻ This shows the SMTP banner line of the remote SMTP server.

❼ This shows the SMTP status code returned for this address; 250 is the basic SMTP success code and in addition, this remote SMTP server responds with extended SMTP status codes and some additional text.

Figure 31–2 Logging: Including Optional Logging Fields

```
15-Nov-2012 19:16:57.64 l            tcp_local    E 1
 adam@example.com rfc822;marlowe@example.com marlowe@example.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKLSKLZLI90N15M.00;1 <01ISKLSKC2QC90N15M@example.com> ❶
```

Figure 31–2 Cont'd on next page

Figure 31–2 (Cont.) Logging: Including Optional Logging Fields

```
15-Nov-2012 19:17:01.16 tcp_local               D 1
 adam@example.com rfc822;marlowe@example.com marlowe@example.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKLSKLZLI90N15M.00 <01ISKLSKC2QC90N15M@example.com> ❷
 dns;thor.example.com (TCP|206.184.139.12|2788|192.160.253.66|25)
 (THOR.EXAMPLE.COM -- Server ESMTP [PMDF V5.1-10 #8694])
 smtp;250 2.1.5 marlowe@example.com and options OK.
```

This example is similar to that shown in Figure 31–1, but with the additional information logged by setting LOG_FILENAME=1 and LOG_MESSAGE_ID=1 showing the filename and message-id; see ❶ and ❷. The message-id in particular can be used to correlate which entries relate to which message.

Figure 31–3 Logging: Sending to a List

```
15-Nov-2012 20:01:44.10 l           l           E 1
 adam@example.com rfc822;test-list@example.com bob
 PMDF_QUEUE:[l]ZZ01ISKND3DE1K90N15M.00;1 <01ISKND2H8MS90N15M@example.com>

15-Nov-2012 20:01:44.81 l           tcp_local   E 1
 adam@example.com rfc822;test-list@example.com carol@otherco.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKND2WS1I90N15M.00;1 <01ISKND2H8MS90N15M@example.com>

15-Nov-2012 20:01:44.81 l           tcp_local   E 1
 adam@example.com rfc822;test-list@example.com david@otherco.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKND2WS1I90N15M.00;1 <01ISKND2H8MS90N15M@example.com>

15-Nov-2012 20:01:50.69 l                       D 1
 adam@example.com rfc822;test-list@example.com bob
 PMDF_QUEUE:[l]ZZ01ISKND3DE1K90N15M.00 <01ISKND2H8MS90N15M@example.com>

15-Nov-2012 20:01:57.36 tcp_local               D 1
 adam@example.com rfc822;test-list@example.com carol@otherco.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKND2WS1I90N15M.00 <01ISKND2H8MS90N15M@example.com>
 dns;gw.otherco.com (TCP|206.184.139.12|2788|192.160.253.66|25)
 (gw.otherco.com -- SMTP Sendmail)
 smtp;250 OK.

15-Nov-2012 20:02:06.14 tcp_local               D 1
 adam@example.com rfc822;test-list@example.com david@otherco.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKND2WS1I90N15M.00 <01ISKND2H8MS90N15M@example.com>
 dns;gw.otherco.com (TCP|206.184.139.12|2788|192.160.253.66|25)
 (gw.otherco.com -- SMTP Sendmail)
 smtp;250 OK.
```

The preceding entries illustrate sending to multiple recipients with LOG_FILENAME=1 and LOG_MESSAGE_ID=1 enabled. Here user adam@example.com has sent to the PMDF mailing list test-list@example.com, which expanded to bob@example.com, carol@otherco.com, and david@otherco.com. Note that the original envelope To: address is test-list@example.com for each recipient, though the current envelope To: address is each respective address. Note how the message-id is the same throughout, though two separate files (one for the L channel and one going out the tcp_local channel) are involved.

Figure 31–4 Logging: Sending to a non-existent Domain

```
15-Nov-2012 20:49:04 l            tcp_local    E 1
 adam@example.com rfc822;user@very.bogus.com user@very.bogus.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKP0S0LVQ94DU0K.00;1 <01ISKP0RYMAS94DU0K@EXAMPLE.COM>

15-Nov-2012 20:49:33 tcp_local    process      E 1 ❶
  rfc822;adam@example.com adam@example.com  ❷
 PMDF_QUEUE:[process]ZZ01ISKP0S0LVQ94DTZB.00
 <01ISKP22MW8894DTAS@EXAMPLE.COM>,<01ISKP0RYMAS94DU0K@EXAMPLE.COM> ❸

15-Nov-2012 20:49:33 tcp_local    process      E 1 ❹
  rfc822;postmaster@example.com postmaster@example.com
 PMDF_QUEUE:[process]ZZ01ISKP0S0LVQ94DTZB.00
 <01ISKP22MW8894DTAS@EXAMPLE.COM>,<01ISKP0RYMAS94DU0K@EXAMPLE.COM>

15-Nov-2012 20:50:07 tcp_local                 R 1 ❺
 adam@example.com rfc822;user@very.bogus.com user@very.bogus.com
 PMDF_QUEUE:[tcp_local]ZZ01ISKP0S0LVQ94DU0K.00 <01ISKP0RYMAS94DU0K@EXAMPLE.COM>
 Illegal host/domain name found ❻

15-Nov-2012 20:50:08 process      l            E 3 ❼
  rfc822;adam@example.com adam ❽
 PMDF_QUEUE:[l]ZZ01ISKP23BUQS94DTYL.00;1 <01ISKP22MW8894DTAS@EXAMPLE.COM>

15-Nov-2012 20:50:08 process      l            E 3
  rfc822:postmaster@example.com postmaster
 PMDF_QUEUE:[l]ZZ01ISKP23BUQS94DTYL.00;1 <01ISKP22MW8894DTAS@EXAMPLE.COM>

15-Nov-2012 20:50:12 l                         D 3
  rfc822;adam@example.com adam
 PMDF_QUEUE:[l]ZZ01ISKP23BUQS94DTYL.00 <01ISKP22MW8894DTAS@EXAMPLE.COM>

15-Nov-2012 20:50:12 l                         D 3
  rfc822;postmaster@example.com postmaster
 PMDF_QUEUE:[l]ZZ01ISKP23BUQS94DTYL.00 <01ISKP22MW8894DTAS@EXAMPLE.COM>
```

The above entries illustrate an attempt to send to a non-existent pseudodomain (here very.bogus.com); that is, sending to a pseudodomain name that is not noticed as illegal by PMDF's rewrite rules and which PMDF matches to an outgoing TCP/IP channel. This example assumes PMDF option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1.

When the TCP/IP channel runs and checks for the pseudodomain name in the DNS, the DNS returns an error that no such name exists. Note the "rejection" entry (R), as seen in ❺, with the DNS returning an error that this is not a legal domain name, as seen in ❻. Since the address is rejected after the message has been submitted, PMDF has to generate a bounce message back to the original sender. Note that PMDF enqueues the new rejection message to the original sender, ❶, and (depending on how PMDF is configured regarding bounce copies to the postmaster, as discussed in Section 2.3.4.21) copied to the postmaster, see ❸, *before* deleting the original outbound message (the R entry shown in ❺). Note that notification messages, such as bounce messages, have an empty envelope From: address—as seen, for instance, in ❷ and ❽ in which the envelope From: field is shown as an empty space. Note that the initial enqueue of a bounce message generated by PMDF shows the message-id for the new notification message

Figure 31–4 Cont'd on next page

Figure 31–4 (Cont.) Logging: Sending to a non-existent Domain

followed by the message-id for the original message, ❸. (Such information is not always available to PMDF, but when it is available to be logged it allows correlation of the log entries corresponding to the outbound failed message with the log entries corresponding to the resulting notification message.) Such notification messages are enqueued to the process channel, which in turn enqueues them to an appropriate destination channel, ❼.

Figure 31–5 Logging: Sending to a non-existent Remote User

```
15-Nov-2012 13:11:05 l            tcp_local    E 1
 adam@example.com rfc822;nonesuch@example.com nonesuch@example.com
 PMDF_QUEUE:[tcp_local]ZZ01ISLNBB1JOE94DUWH.00;1 <01ISLNBAWV3094DUWH@example.com>

15-Nov-2012 13:11:08 tcp_local    process      E 1
  rfc822;adam@example.com adam@example.com
 PMDF_QUEUE:[process]ZZ01ISLNBB1JOE94DSGB.00;1
 <01ISLNBFKIDS94DUJ8@example.com>,<01ISLNBAWV3094DUWH@example.com>

15-Nov-2012 13:11:08 tcp_local    process      E 1
  rfc822;postmaster@example.com postmaster@example.com
 PMDF_QUEUE:[process]ZZ01ISLNBB1JOE94DSGB.00;1
 <01ISLNBFKIDS94DUJ8@example.com>,<01ISLNBAWV3094DUWH@example.com>

15-Nov-2012 13:11:11 tcp_local                 R 1 ❶
 adam@.example.com rfc822;nonesuch@example.com nonesuch@example.com
 PMDF_QUEUE:[tcp_local]ZZ01ISLNBB1JOE94DUWH.00 <01ISLNBAWV3094DUWH@example.com>
 dns;thor.example.com (TCP|206.184.139.12|2788|192.160.253.66|25) ❷
 (THOR.EXAMPLE.COM -- Server ESMTP [PMDF V5.1-10 #8694])
 smtp; 553 unknown or illegal user: nonesuch@example.com ❸

15-Nov-2012 13:11:12 process      l            E 3
  rfc822;adam@example PMDF_QUEUE:[l]ZZ01ISLNBGND1094DQDP.00;1
  <01ISLNBFKIDS94DUJ8@example.com>

15-Nov-2012 13:11:12 process      l            E 3
  rfc822;postmaster@example.com postmaster
 PMDF_QUEUE:[l]ZZ01ISLNBGND1094DQDP.00;1 <01ISLNBFKIDS94DUJ8@example.com>

15-Nov-2012 13:11:13 l                         D 3
  rfc822;adam@example.com adam@example.com
 PMDF_QUEUE:[l]ZZ01ISLNBGND1094DQDP.00 <01ISLNBFKIDS94DUJ8@example.com>

15-Nov-2012 13:11:13 l                         D 3
  rfc822;postmaster@example.com postmaster@example.com
 PMDF_QUEUE:[l]ZZ01ISLNBGND1094DQDP.00 <01ISLNBFKIDS94DUJ8@example.com>
```

The above entries illustrate an attempt to send to a bad address on a remote system. This example assumes PMDF option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1, and channel option settings of LOG_BANNER=1 and LOG_TRANSPORTINFO=1. Note the rejection entry (R), seen in ❶. But in contrast to the rejection entry in Figure 31–4, note that the rejection entry here shows that a connection to a remote system was made, and shows the SMTP error code issued by the remote

Figure 31–5 Cont'd on next page

Figure 31–5 (Cont.) Logging: Sending to a non-existent Remote User

SMTP server, ❷ and ❸. The inclusion of the information shown in ❷ is due to setting the channel options LOG_BANNER=1 and LOG_TRANSPORTINFO=1

Figure 31–6 Logging: Rejecting a Remote Side's Attempt to Submit a Message

```
15-Nov-2012 12:02:23 tcp_local                J 0 ❶
 harold@hotmail.com rfc822; alan@very.bogus.com  ❷
 550 5.7.1 Relaying not permitted: alan@very.bogus.com ❸
```

The above entry illustrates the sort of log file entry resulting when PMDF rejects a remote side's attempt to submit a message. (This example assumes that no optional LOG_* options are enabled, so only the basic fields are logged in the entry. Note that enabling the LOG_CONNECTION option, in particular, would result in additional informative fields in such J entries.) In this case, the example is for a site has set up SMTP relay blocking (see Section 16.1.6) with an ORIG_SEND_ACCESS mapping including

```
ORIG_SEND_ACCESS

! ...numerous entries omitted...
!
  tcp_local|*|tcp_local|*   $NRelaying$ not$ permitted
```

and where alan@very.bogus.com is *not* an internal address. Hence the attempt of the remote user harold@hotmail.com to relay through the PMDF system to the remote user alan@very.bogus.com is rejected.

❶ This shows the date and time of PMDF rejecting a remote side's attempt to submit a message. Note that as opposed to cases where a PMDF channel is attempting to send a message which is rejected (indicated by R records, as in Figure 31–4 and Figure 31–5 above), this case of PMDF rejecting a message submission attempt is indicated by a J record.

❷ The attempted envelope From: and To: addresses are shown. In this case, no original envelope To: information was available so that field is empty.

❸ The entry includes the SMTP error message PMDF issued to the remote (attempted sender) side.

Figure 31–7 Logging: Multiple Delivery Attempts

```
15-Nov-2012 10:31:05.18 tcp_internal   tcp_local  E 3 ❶
 adam@sample.example.com rfc822;user@some.org user@some.org
 PMDF_QUEUE:[tcp_local]ZZ01IS3D2ZP7FQ9UN54R.00 <01IRUD7SVA3Q9UN2D4@example.com>

15-Nov-2012 10:31:10.37 tcp_local                Q 0 ❷
 adam@sample.example.com rfc822;user@some.org user@some.org
 PMDF_QUEUE:[tcp_local]ZZ01IS3D2ZP7FQ9UN54R.00 <01IRUD7SVA3Q9UN2D4@example.com> ❸
 TCP active open: Failed connect()    Error: no route to host ❹
```

Figure 31–7 Cont'd on next page

Figure 31–7 (Cont.) Logging: Multiple Delivery Attempts

```
...several hours worth of entries...

15-Nov-2012 12:45:39.48 tcp_local                Q 0 ❺
 adam@sample.example.com rfc822;user@some.org user@some.org
 PMDF_QUEUE:[tcp_local]ZY01IS3D2ZP7FQ9UN54R.00 <01IRUD7SVA3Q9UN2D4@example.com> ❻
 TCP active open: Failed connect()    Error: no route to host

...several hours worth of entries...

15-Nov-2012 16:45:24.72 tcp_local                Q 0
 adam@sample.example.com rfc822;user@some.org user@some.org
 PMDF_QUEUE:[tcp_local]ZX01IS67NY4RRK9UN7GP.00 <01IRUD7SVA3Q9UN2D4@example.com> ❼
 TCP active open: Failed connect()    Error: connection refused ❽

...several hours worth of entries...

15-Nov-2012 20:45:51.55 tcp_local                D 3 ❾
 adam@sample.example.com rfc822;user@some.org user@some.org
 PMDF_QUEUE:[tcp_local]ZX01IS67NY4RRK9UN7GP.00 <01IRUD7SVA3Q9UN2D4@example.com>
 dns;host.some.org (TCP|206.184.139.12|2788|192.1.1.1|25)
 (All set, fire away)
 smtp; 250 Ok
```

The above entries illustrate the sort of log file entries resulting when a message can not be delivered upon the first attempt, so that PMDF has to retry sending it several times. This example assumes option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1.

❶ The message comes in the tcp_internal channel—perhaps from a POP or IMAP client, or perhaps from another host within the organization using PMDF as an SMTP relay—and PMDF enqueues it to the outgoing tcp_local channel.

❷ The first delivery attempt fails—this is a Q entry—and note that message size is shown as 0 for such unsuccessful dequeue attempts.

❸ That this is a first delivery attempt can be seen from the ZZ* filename.

❹ This delivery attempt failed when the TCP/IP package could not find a route to the remote side. Note that, as opposed to Figure 31–4, the DNS did not object to the destination domain name, some.org; rather, the "no route to host" error indicates that there is some network problem between the sending and receiving side.

❺ The next time the PMDF periodic job runs it reattempts delivery, again unsuccessfully.

❻ Note that the file name is now ZY*, indicating that this is a second attempt.

❼ Note that the file name is ZX* for this third unsuccessful attempt.

❽ And again the next time the periodic job reattempts delivery the delivery fails, though this time the TCP/IP package is not complaining that it cannot get through to the remote SMTP server, but rather the remote SMTP server is not accepting connections. (Perhaps the remote side fixed their network problem, but has not yet brought their SMTP server back up—or their SMTP server is swamped handling other messages and hence was not accepting connections at the moment PMDF tried to connect.)

Figure 31–7 Cont'd on next page

Figure 31–7 (Cont.) Logging: Multiple Delivery Attempts

❾ Finally the message is dequeued.

Figure 31–8 Logging: Z Entries

```
15-Nov-2012 20:56:43 l            tcp_local    E 1 ❶
 adam@example.com rfc822;barbara@else.where.com barbara@else.where.com
 PMDF_QUEUE:[tcp_local]ZZ01IT1GSE6O069AMK60.00;1 <01IT1GSE4VYC9AMK60@EXAMPLE.COM>

15-Nov-2012 20:56:43 l            tcp_local    E 1 ❷
 adam@example.com rfc822;carl@else.where.com carl@else.where.com
 PMDF_QUEUE:[tcp_local]ZZ01IT1GSE6O069AMK60.00;1 <01IT1GSE4VYC9AMK60@EXAMPLE.COM>

15-Nov-2012 20:56:48 l            tcp_local    E 1 ❸
 adam@example.com rfc822;barbara@else.where.com barbara@else.where.com
 PMDF_QUEUE:[tcp_local]ZZ01IT1GTR1SS69AMJBD.00;1 <01IT1GSE4VYC9AMK60@EXAMPLE.COM> ❹

15-Nov-2012 20:56:48 tcp_local                 Z 1 ❺
 adam@example.com rfc822;barbara@else.where.com barbara@else.where.com
 PMDF_QUEUE:[tcp_local]ZZ01IT1GSE6O069AMK60.00 <01IT1GSE4VYC9AMK60@EXAMPLE.COM>
 smtp;422 user over quota; cannot receive new mail: barbara@else.where.com

15-Nov-2012 20:56:48 tcp_local                 D 1 ❻
 adam@example.com rfc822;carl@else.where.com carl@else.where.com
 PMDF_QUEUE:[tcp_local]ZZ01IT1GSE6O069AMK60.00 <01IT1GSE4VYC9AMK60@EXAMPLE.COM>
 smtp;250 2.1.5 carl@else.where.com and options OK.

15-Nov-2012 20:56:50 tcp_local                 Q 1 ❼
 adam@example.com rfc822;barbara@else.where.com barbara@else.where.com
 PMDF_QUEUE:[tcp_local]ZZ01IT1GTR1SS69AMJBD.00 <01IT1GSE4VYC9AMK60@EXAMPLE.COM>
 barbara@else.where.com:
 smtp;422 user over quota; cannot receive new mail: barbara@else.where.com ❽

...several hours of entries...

15-Nov-2012 23:20:14 tcp_local                 D 1 ❾
 adam@example.com rfc822;barbara@else.where.com barbara@else.where.com
 PMDF_QUEUE:[tcp_local]ZZY1IT1GTR1SS69AMJBD.00 <01IT1GSE4VYC9AMK60@EXAMPLE.COM>
 smtp;250 OK.
```

The above entries illustrate the case of a message file including multiple recipients for which one recipient succeeds and another recipient gets a temporary failure; this example assumes option settings of LOG_FILENAME=1 and LOG_MESSAGE_ID=1. This is a less common case—it can only arise with certain protocols, for instance, SMTP and DECnet MAIL-11, that allow both for multiple recipients per transaction and for temporary failures. When a temporary failure occurs on a message file which had other, successful recipients, PMDF must make a new message file containing only the unsuccessful recipient(s). The original message file (containing all recipients) is deleted. PMDF then immediately retries delivery to the unsuccessful recipient(s) since such temporary errors may be due simply to the remote side not wanting to accept that many recipients all at once. Thus in the above entries, adam@example.com is

Figure 31–8 Cont'd on next page

Figure 31–8 (Cont.) Logging: Z Entries

trying to send to two recipients at the same remote site, barbara@else.where.com, and carl@else.where.com.

❶ We see the initial enqueue to the first recipient...

❷ ...and the initial enqueue to the second recipient; note that the message id is the same, since it is the same message to both recipients, and indeed the file name is the same meaning both recipients will be attempted in the same SMTP transaction.

❸ Here we see PMDF reenqueuing a new message file containing only the unsuccessful message recipient. The entry showing the unsuccessful dequeue attempt is written later, in ❺—PMDF cannot delete the original message file until the new message file is written, so this reenqueue operation is completed and its log entry written before the unsuccessful dequeue attempt log entry can be written.

❹ Note that the message id for the reenqueue is the same (this is the same message content) as in the original enqueue shown in ❶, but the file name is different (it is a different file since this new file contains only barbara@else.where.com as a recipient).

❺ For the barbara@else.where.com recipient address, the remote side issued a temporary error. PMDF does not delete (dequeue) the original message file (containing both recipients) until after the new message file (containing only the barbara@else.where.com recipient) has been written (reenqueued); that is why the ❸ log entry for the reenqueue for the barbara@else.where.com recipient appears *before* the ❺ and ❻ log entries for the dequeue of the original message file.

❻ The carl@else.where.com recipient address was accepted by the remote side.

❼ Since the failure on the barbara@else.where.com recipient was a temporary failure on a message where another recipient was accepted, PMDF immediately retries delivery to the unsuccessful recipient.

❽ This immediate retry is again unsuccessful, (as indicated by being a "Q" record ❼), with an error message as shown here. So the message will await retry by the next periodic delivery job.

❾ This example shows a case where the remote barbara@else.where.com user apparently cleared out some disk space; the next retry succeeded.

Figure 31–9 Logging: Incoming SMTP Message Routed Through the Conversion Channel

```
15-Nov-2012 00:06:26.72 tcp_local    conversion   E 9 ❶
 amy@example.com rfc822;bert@example.com bert@example.com
 PMDF_QUEUE:[conversion]ZZ01IT5UAMZ4QW98518O.00;1 <01IT5UALL14498518O@state.edu>

15-Nov-2012 00:06:29.06 conversion   l            E 9 ❷
 amy@sample.com rfc822;bert@example.com bert
 PMDF_QUEUE:[l]ZZ01IT5UAOXLDW98509E.00;1 <01IT5STUMUFO984Z8L@sample.com>

15-Nov-2012 00:06:29.31 conversion                D 9 ❸
 amy@sample.com rfc822;bert@example.com bert
 PMDF_QUEUE:[conversion]ZZ01IT5UAMZ4QW98518O.00 <01IT5UALL14498518O@sample.com>

15-Nov-2012 00:06:32.62 l                         D 9 ❹
 amy@sample.com rfc822;bert@example.com bert
 PMDF_QUEUE:[l]ZZ01IT5UAOXLDW98509E.00 <01IT5STUMUFO984Z8L@sample.com>
```

The above entries illustrate the case of a message routed through the `conversion` channel. That is, this is an example where the site is assumed to have a `CONVERSION` mapping table along the lines of

```
CONVERSIONS

  IN-CHAN=tcp_local;OUT-CHAN=l;CONVERT    Yes
```

This example assumes option settings of `LOG_FILENAME=1` and `LOG_MESSAGE_ID=1`.

❶ The message from external user amy@sample.com comes in addressed to the `L` channel recipient bert@example.com. The `CONVERSIONS` mapping entry, however, causes the message to be initially enqueued to the `conversion` channel (rather than directly to the `L` channel).

❷ The `conversion` channel runs and enqueues the message to the L channel.

❸ Then the `conversion` channel can dequeue the message (delete the old message file).

❹ And finally the `L` channel dequeues (delivers) the message.

Figure 31–10 Logging: Outbound Connection Logging

```
15-Nov-2012 10:52:05.41 1e488.0 l            tcp_local    E 1
 adam@example.com rfc822;bobby@sample.example.com bobby@sample.example.com
 PMDF_QUEUE:[tcp_local]ZZ01ITRF7BO388000FCN.00;1 <01ITRF7BDHS6000FCN@EXAMPLE.COM> ❶

15-Nov-2012 10:52:05.41 1e488.0 l            tcp_local    E 1
 adam@example.com rfc822;carl@sample.example.com carl@sample.example.com
 PMDF_QUEUE:[tcp_local]ZZ01ITRF7BO388000FCN.00;1 <01ITRF7BDHS6000FCN@EXAMPLE.COM> ❷

15-Nov-2012 10:52:05.74 1e488.1 l            tcp_local    E 1
 adam@example.com rfc822;dave@milan.example.com dave@milan.example.com
 PMDF_QUEUE:[tcp_local]ZZ01ITRF7C11FU000FCN.00;1 <01ITRF7BDHS6000FCN@EXAMPLE.COM> ❸
```

Figure 31–10 Cont'd on next page

Figure 31–10 (Cont.) Logging: Outbound Connection Logging

```
15-Nov-2012 10:52:10.79 1f625.2.0 tcp_local    -            O ❹
 TCP|206.184.139.12|5900|206.184.139.66|25
 SMTP/milan.example.com/mailhub.example.com ❺

15-Nov-2012 10:52:10.87 1f625.3.0 tcp_local    -            O ❻
 TCP|206.184.139.12|5901|206.184.139.70|25
 SMTP/sample.example.com/sample.example.com ❼

15-Nov-2012 10:52:12.28 1f625.3.1 tcp_local                 D 1
 adam@example.com rfc822;bobby@sample.example.com bobby@sample.example.com
 PMDF_QUEUE:[tcp_local]ZZ01ITRF7BO388000FCN.00 <01ITRF7BDHS6000FCN@EXAMPLE.COM>
 sample.example.com dns;sample.example.com ❽
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 (sample.example.com -- Server ESMTP [PMDF V5.1-9 #8790])
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 smtp;250 2.1.5 bobby@sample.example.com and options OK.

15-Nov-2012 10:52:12.28 1f625.3.1 tcp_local                  D 1
 adam@example.com rfc822;carl@sample.example.com carl@sample.example.com
 PMDF_QUEUE:[tcp_local]ZZ01ITRF7BO388000FCN.00 <01ITRF7BDHS6000FCN@EXAMPLE.COM>
 sample.example.com dns;sample.example.com
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 (sample.example.com -- Server ESMTP [PMDF V5.1-9 #8790])
 (TCP|206.184.139.12|5901|206.184.139.70|25)
 smtp;250 2.1.5 carl@sample.example.com and options OK.

15-Nov-2012 10:52:12.40 1f625.3.2 tcp_local     -            C ❾
 TCP|206.184.139.12|5901|206.184.139.70|25
 SMTP/sample.example.com/sample.example.com

15-Nov-2012 10:52:13.01 1f625.2.1 tcp_local                  D 1
 adam@example.com rfc822;dave@milan.example.com dave@milan.example.com
 PMDF_QUEUE:[tcp_local]ZZ01ITRF7C11FU000FCN.00 <01ITRF7BDHS6000FCN@EXAMPLE.COM>
 mailhub.example.com dns;mailhub.example.com
 (TCP|206.184.139.12|5900|206.184.139.66|25)
 (MAILHUB.EXAMPLE.COM -- Server ESMTP [PMDF V5.1-7 #8694])
 (TCP|206.184.139.12|5900|206.184.139.66|25)
 smtp;250 2.1.5 dave@milan.example.com and options OK.

15-Nov-2012 10:52:13.05 1f625.2.2 tcp_local     -            C ❿
 TCP|206.184.139.12|5900|206.184.139.66|25
 SMTP/milan.example.com/mailhub.example.com
```

The above entries illustrate log output for an outgoing message when connection logging is enabled, via `LOG_CONNECTION=3`. `LOG_PROCESS=1`, `LOG_MESSAGE_ID=1`, and `LOG_FILENAME=1` are also assumed in this example. The example shows the case of user adam@example.com sending the same message (note that the message ID is the same for each message copy) to three recipients, bobby@sample.example.com, carl@sample.example.com, and dave@milan.example.com. This example assumes that the message is going out a `tcp_local` channel marked (as such channels usually are) with the `single_sys` channel keyword. Therefore, a separate message file on disk will be created for each set of recipients to a separate host name, as seen in ❶, ❷, and ❸, where the bobby@sample.example.com and carl@sample.example.com recipients are

Figure 31–10 Cont'd on next page

Figure 31–10 (Cont.) Logging: Outbound Connection Logging

stored in the same message file, but the dave@milan.example.com recipient is stored in a different message file.

❶ The message is enqueued to the first recipient...

❷ ...and to the second recipient...

❸ ...and to the third recipient.

❹ Having `LOG_CONNECTION=3` set causes PMDF to write this entry. The minus,–, indicates that this entry refers to an outgoing connection. The O means that this entry corresponds to the opening of the connection. Also note that the process id here is the same, `1f625`, as in ❻, since the same process is used for the multithreaded TCP/IP channel for these separate connection opens, though this open is being performed by thread 2 *vs.* thread 3 for ❻.

❺ As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each – the first in this entry, and the second shown in ❼. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. That is, in the SMTP/*initial-host/dns-host* clauses, note the display of both the initial host name, and that used after performing a DNS MX record lookup on the initial host name: the system milan.example.com is apparently MXed to mailhub.example.com.

❻ The multithreaded SMTP client opens up a connection to the second system in a separate thread (though the same process).

❼ As there are two separate remote systems to which to connect, the multithreaded SMTP client in separate threads opens up a connection to each – the second in this entry, and the first shown above in ❺. This part of the entry shows the sending and destination IP numbers and port numbers, and shows both the initial host name, and the host name found by doing a DNS lookup. In this example, the system sample.example.com apparently receives mail directly itself.

❽ Besides resulting in specific connection entries, `LOG_CONNECTION=3` also causes inclusion of connection related information in the regular message entries, as seen here for instance.

❾ Having LOG_CONNECTION=3 causes PMDF to write this entry. After any messages are dequeued, (the bobby and carl messages in this example), the connection is closed, as indicated by the C in this entry.

❿ Having LOG_CONNECTION=3 causes PMDF to write this entry. After any messages are dequeued, (the dave message in this example), the connection is closed, as indicated by the C in this entry.

Figure 31–11 Logging: Inbound Connection Logging

Figure 31–11 Cont'd on next page

Figure 31–11 (Cont.) Logging: Inbound Connection Logging

```
15-Nov-2012 17:02:08.70 tcp_local    +            O    ❶
 TCP|206.184.139.12|25|192.160.253.66|1244 SMTP       ❷

15-Nov-2012 17:02:26.65 tcp_local    l            E 1
 service@example.com rfc822;adam@example.com adam
 THOR.EXAMPLE.COM (THOR.EXAMPLE.COM [108.165.158.93]) ❸

15-Nov-2012 17:02:27.05 tcp_local    +            C    ❹
 TCP|206.184.139.12|25|192.160.253.66|1244 SMTP

15-Nov-2012 17:02:31.73 l                         D 1
 service@example.com rfc822;adam@example.com adam
```

The above entries illustrate log output for an incoming SMTP message when connection logging is enabled, via `LOG_CONNECTION=3`.

❶ The remote system opens a connection. The O character indicates that this entry regards the "O"pening of a connection; the + character indicates that this entry regards an incoming connection.

❷ The IP numbers and ports for the connection are shown. In this entry, the receiving system (the system making the log file entry) has IP address 206.184.139.12 and the connection is being made to port 25; the sending system has IP address 192.160.253.66 and is sending from port 1244.

❸ In the entry for the enqueue of the message from the incoming TCP/IP channel (tcp_local) to the L channel recipient, note that information beyond the default is included since LOG_CONNECTION=3 is enabled. Specifically, the name that the sending system claimed on its HELO or EHLO line, the sending system's name as found by a DNS reverse lookup on the connection IP number, and the sending system's IP address are all logged; see Section 2.3.4.40 for a discussion of channel keywords affecting this behavior.

❹ The inbound connection is closed. The C character indicates that this entry regards the "C"losing of a connection; the + character indicates that this entry regards an incoming connection.

31.2 Web-based QM Utility

The web-based QM utility is a facility for managing the PMDF message queues. It allows inspection and manipulation of queued messages. In these capabilities the web-based QM utility is comparable to the command line PMDF QM (see Section 29.2.1) (OpenVMS) or `pmdf qm (see Section 30.2.2)` (UNIX and NT) utility. However, the web-based QM utility also allows for stopping or running PMDF processing jobs.

31.2.1 Accessing the Web-based QM Utility

In order for the web-based QM utility to be available and accessible the following items are necessary:

1. The PMDF Dispatcher must be configured to run the PMDF HTTP server.
2. The PMDF HTTP server must be configured to know about the QM CGI.
3. An HTTP_ACCESS mapping table allowing access to the QM utility must be present in the PMDF mappings file.

The best way to achieve the above configuration items is to run the web-based PMDF configuration utility or the command line Dispatcher configuration utility to generate the needed files. For manual configuration, see Chapter 11 and Chapter 12.

If enabled, the web-based QM utility is available via the URL:

```
http://hostname:7633/qm/
```

where *hostname* is the TCP/IP name of your PMDF system.

In order to use the web-based QM utility, you will need to authenticate yourself as one of: the SYSTEM or PMDF account on OpenVMS, the `root` or `pmdf` account on UNIX, or the `Administrator` account on NT.

31.2.2 Examples of the Web-based QM Utility's Web Page Displays

This section shows examples of the web-based QM utility's web page displays on a sample system. Figure 31–12 shows the initial page displayed at

```
http://hostname:7633/qm/
```

From this initial page, you may request either a full or quick listing of message files currently in PMDF channel queues. The requested listing will not be displayed unless and until you have authenticated yourself as a PMDF manager (that is, as SYSTEM or PMDF on OpenVMS, or as `root` or `pmdf` on UNIX, or as `Administrator` on NT).

Figure 31–12 Web-based QM Home Page

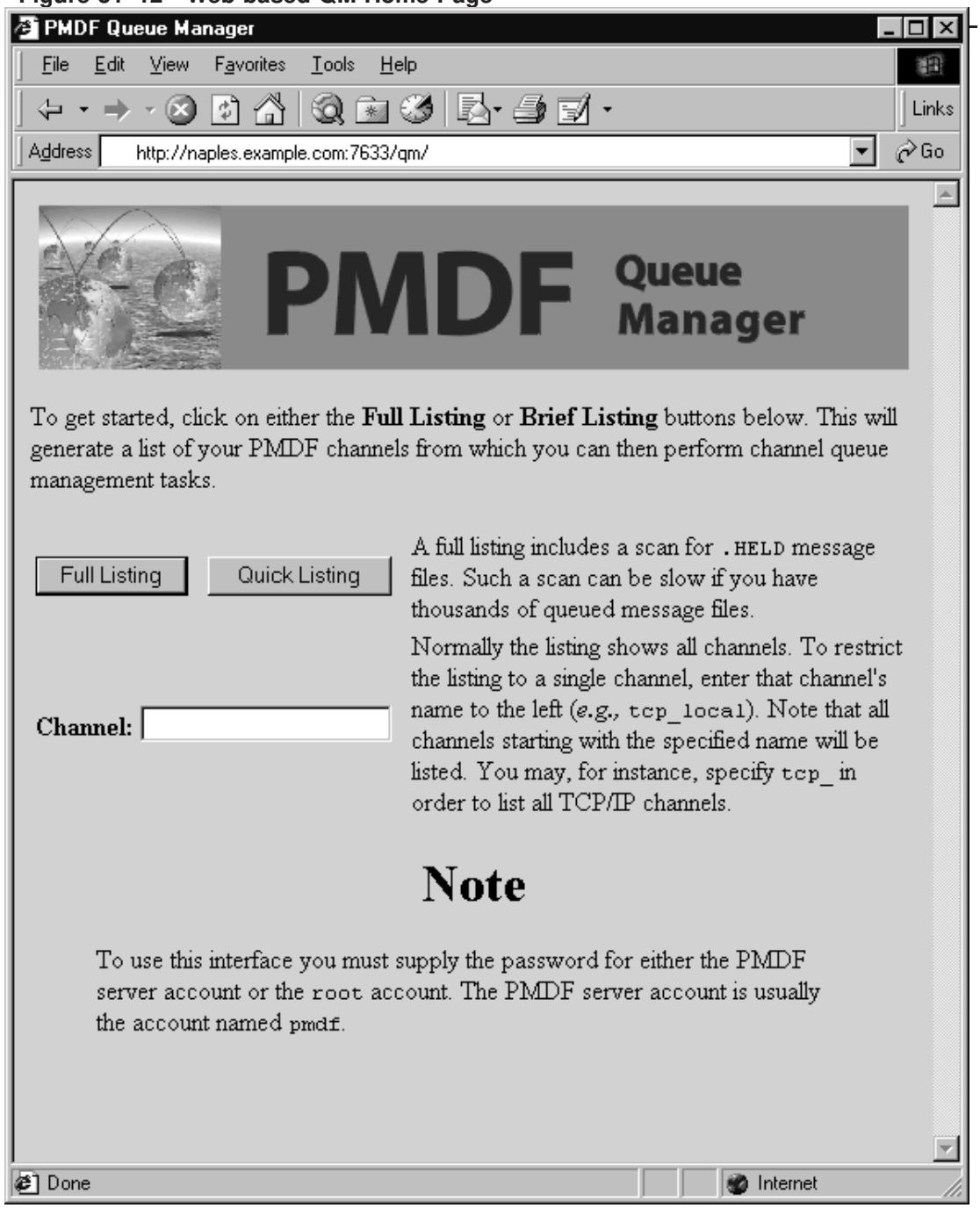


Figure 31–13 shows a sample of a quick listing, as displayed by selecting the "Quick Listing" button from Figure 31–12.

Figure 31–13 Web-based QM Quick Listing Page

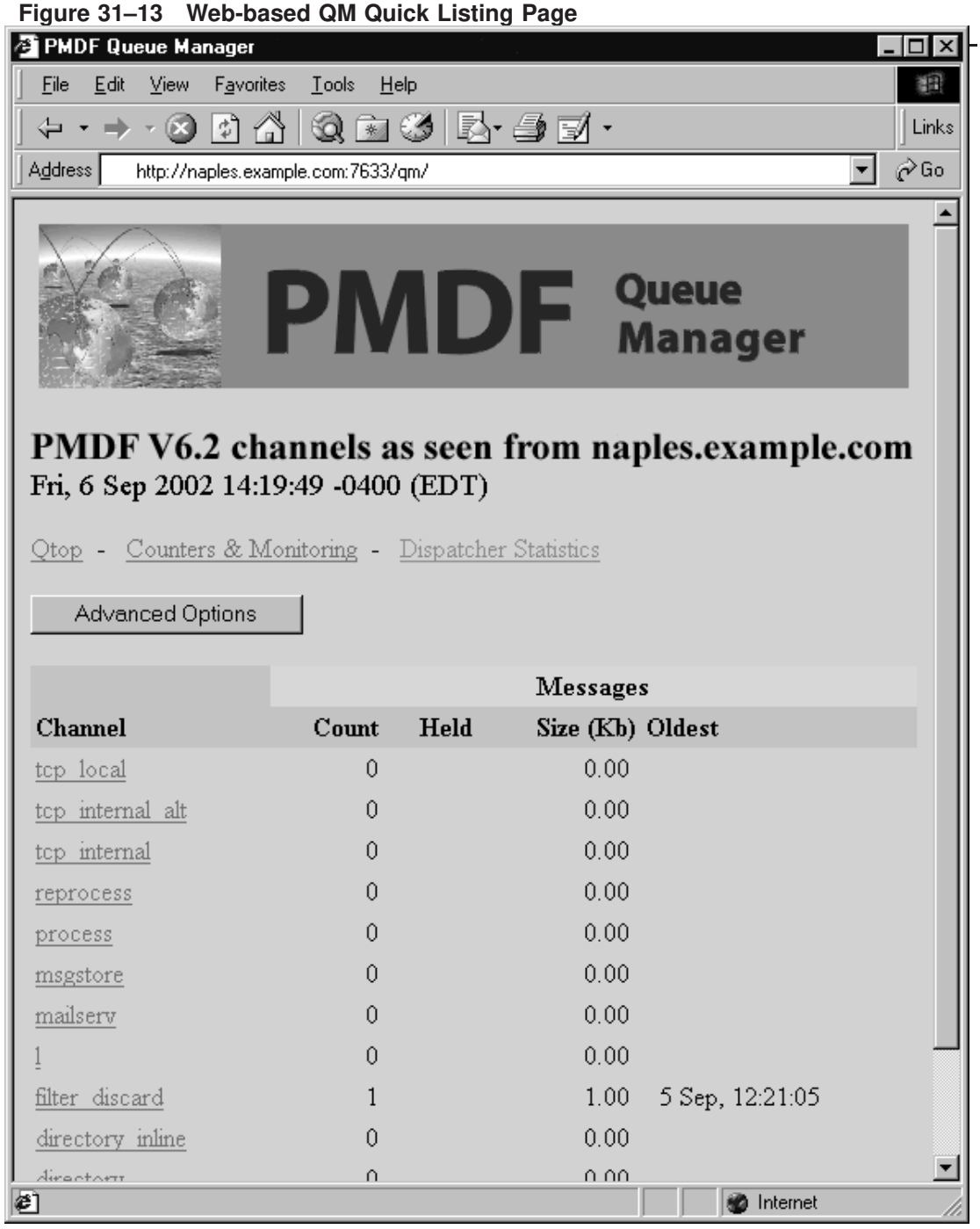


Figure 31–14 shows a sample of an advanced options page, as displayed by selecting the “Advanced Options” button from Figure 31–13. Scrolling further down such a page past the point visible in this figure, you would be presented with buttons for each channel for displaying the messages queued for that channel, for submitting a processing job for that channel, or for stopping that channel from running any new processing jobs.

Figure 31–14 Web-based QM Advanced Options Page

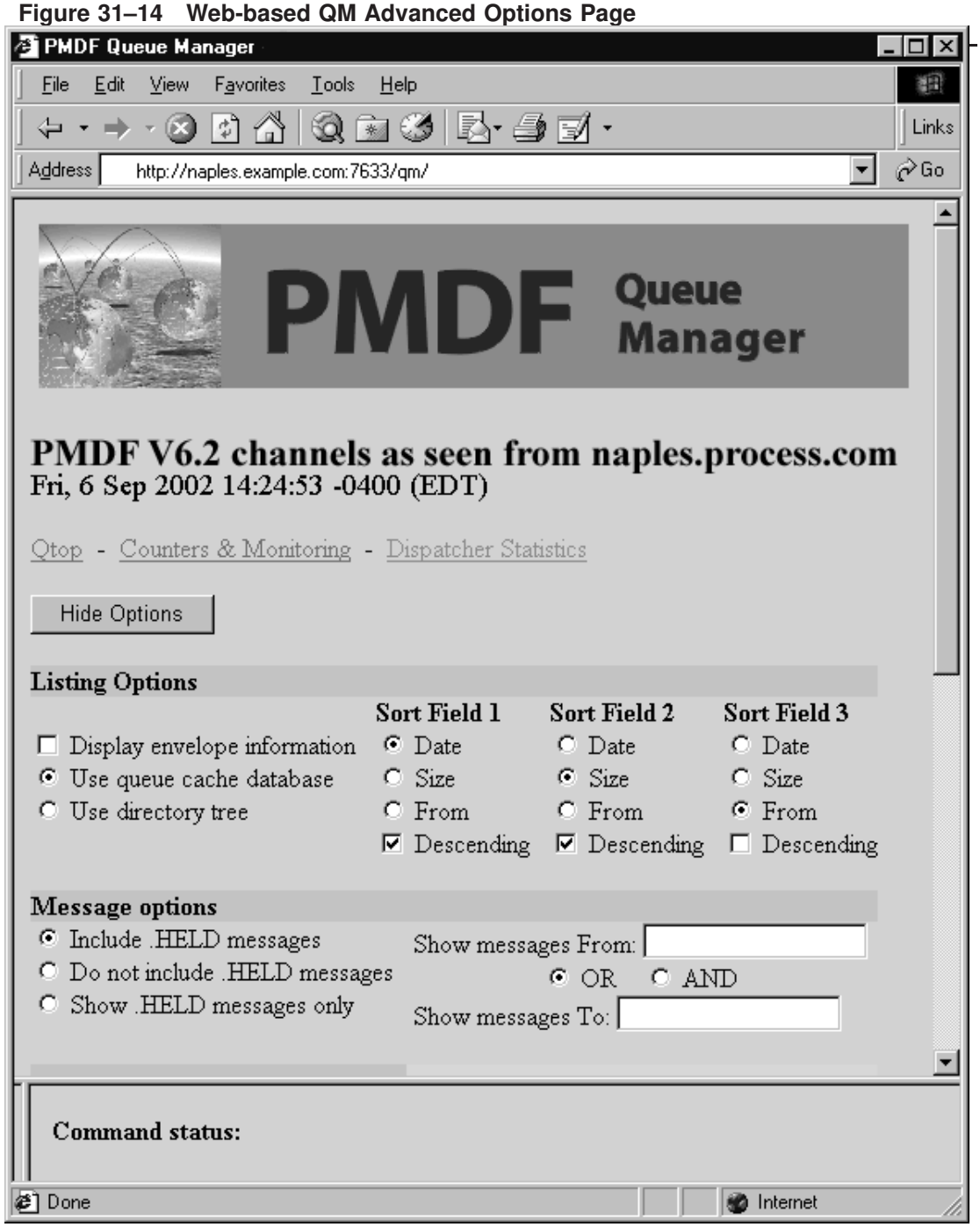


Figure 31–15 shows a sample of the `filter_discard` channel display page, as displayed by clicking on the filter_discard channel from the quick listing page, Figure 31–13, or by selecting the `List` button from the advanced options page.

Figure 31–15 Web-based QM I Channel Page

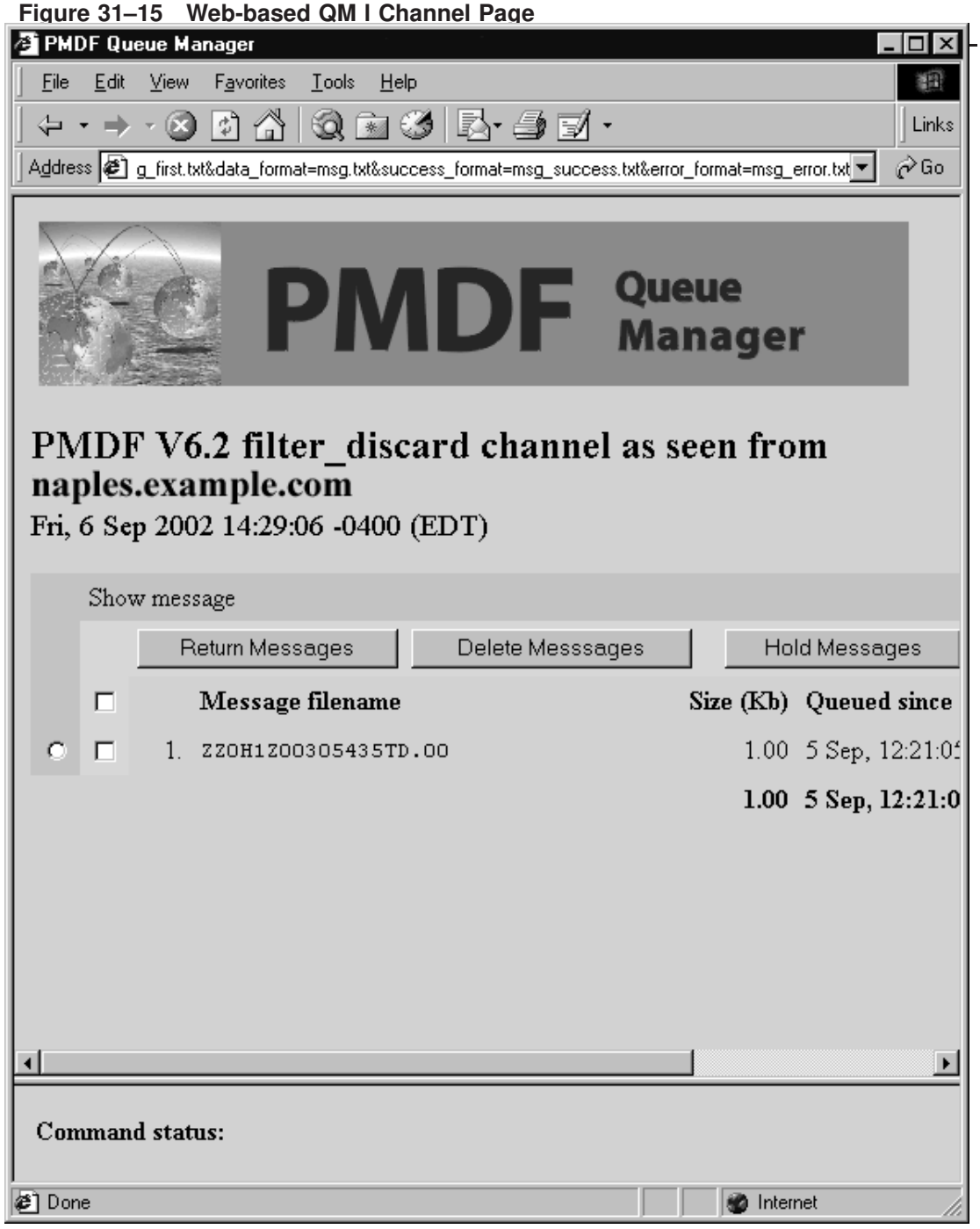
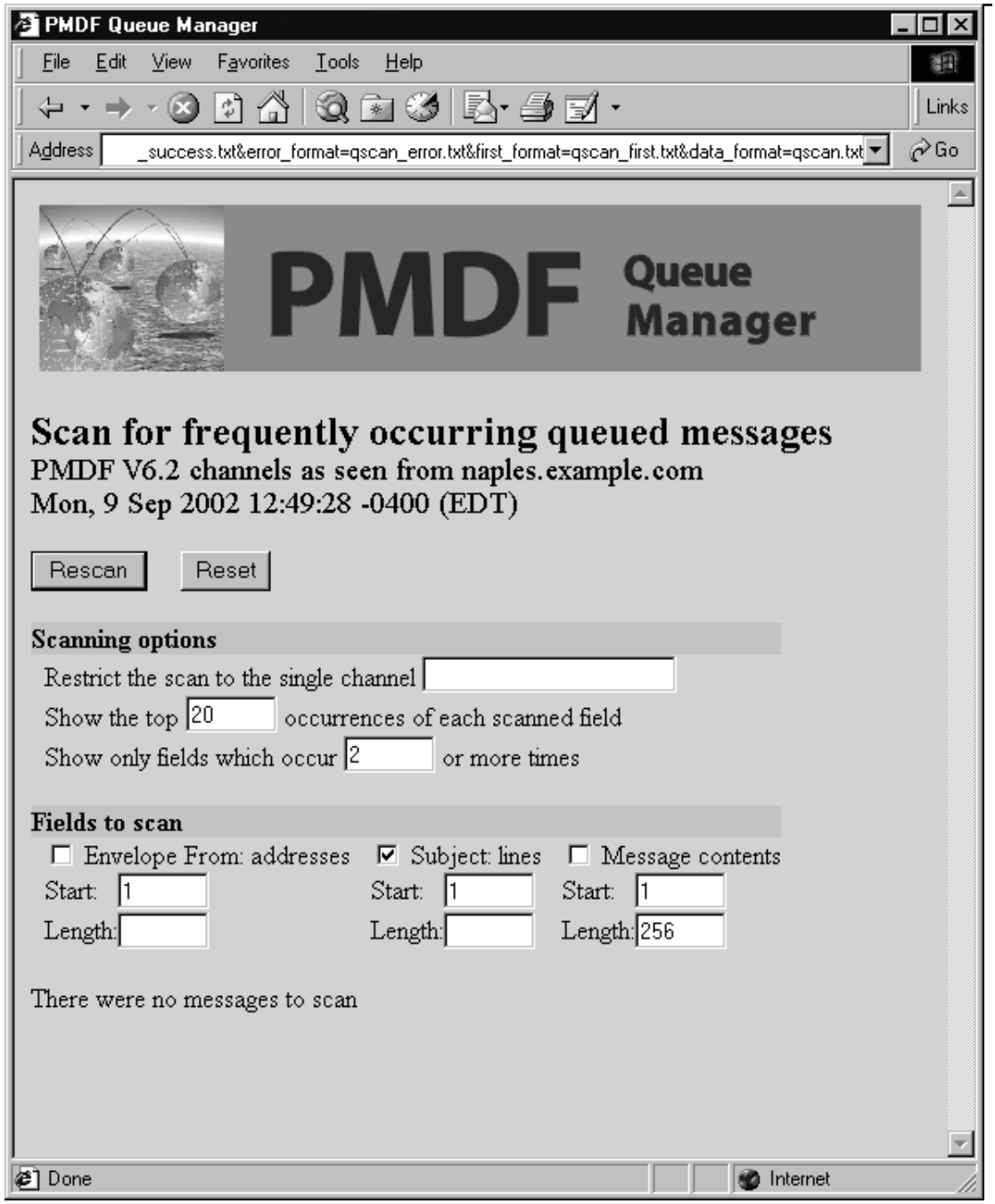


Figure 31–16 shows a sample of the Qtop page, as displayed by clicking on “Qtop” from Figure 31–13 or Figure 31–14. Qtop may be used to see what strings or addresses are frequently appearing in messages currently in the PMDF queue area. This can be particularly useful for spotting flurries of unsolicited bulk e-mail (spam) or chain letters. Figure 31–16 shows an example where there are numerous messages in the PMDF queue area whose Subject: header field is “Relay test”.

Figure 31–16 Web-based QM Qtop Page



The web-based QM utility also has links on the listing and advanced options pages to the Dispatcher statistics display, such as the sample display shown in Figure 11–5.

31.3 Message Circuit Checking

The PMDF message circuit checking facility may be used to monitor the health of the PMDF system and of other e-mail connected systems, by periodically looping messages through (back to the circuit check facility) and monitoring on the delivery time for the loop to complete.

The PMDF manager configures the facility to periodically send messages to loopback addresses (addresses that will loop through PMDF channels or remote systems and come back to the circuit checking facility). The sizes and other aspects of such messages may be controlled. A circuit check detached process runs, that wakes up periodically and sends out circuit messages and looks for which circuit messages have arrived back. And the facility maintains binned counters for the lengths of time the messages take to complete their circuit. Thus the components are:

- The circuit check configuration file, controlling what sorts of messages are sent where and when.
- A circuit check channel definition in the PMDF configuration file, and an alias pointing to the circuit check channel address.
- The circuit check detached process.
- The database of circuit completion counters.
- The PMDF CIRCUIT_CHECK/SHOW (OpenVMS) or `pmdf circuit_check -show` (UNIX or NT) utility which may be used to display the values in the circuit completion counters database.

These components will be discussed further in the subsections below.

31.3.1 Configuring the Message Circuit Check Facility

The first step in installing the message circuit check facility is to insert the channel definition in the PMDF configuration file. The channel definition should have the form:

```
circuitcheck slave
CIRCUITCHECK-DAEMON
```

Note that such a channel should always be marked with the `slave` keyword. (Indeed, there is no actual channel program executed, not even a slave channel program; all the actual circuit check operation is performed by the circuit check detached process.)

An alias entry should also be added to the PMDF alias file. Pick a name, *e.g.* “circuitcheck”, that is otherwise unused on your system, and then add an entry such as:

```
circuitcheck:  circuitcheck@CIRCUITCHECK-DAEMON
```

This will set up the address circuitcheck@*your-local-domain*, where *your-local-domain* is the official host name of the local channel, to be the loopback address—the eventual destination address to use when sending circuit check messages.

An option file can also be specified, if desired. This file should be located in the PMDF table directory and named `circuitcheck_option.` on OpenVMS or `circuitcheck_option` on UNIX or NT. One available option is:

INTERVAL (60 <= integer <= 3600)
This option specifies in seconds how frequently the circuit check detached process should "wake up" to look for received circuit check messages and, if an appropriate interval has elapsed, send new circuit check messages. The default if this option is not specified is 240, corresponding to four minutes.

Once the channel and alias are set up, the next step is to configure the message circuits and timings of your choice in the circuit check configuration file, described below in Section 31.3.1.1.

Note: Make sure you do **not** have the `routelocal` channel keyword on the channels used for the systems you want to monitor using the circuit check facility.

31.3.1.1 The Circuit Check Configuration File

The actual circuit messages sent by the circuit check facility are controlled by rules in the circuit check configuration file. This file must be located in the PMDF table directory and named `circuitcheck.cnf`.

Note that the circuit check configuration file is part of a compiled configuration, so if using a compiled PMDF configuration you must recompile (and on OpenVMS reinstall the compiled configuration) after changes to the file. The circuit check detached process will not see changes to its configuration file until it is restarted via a command such as PMDF RESTART CIRCUIT_CHECK (OpenVMS) or `pmdf restart circuit_check` (UNIX and Windows).

The circuit check configuration file is a text file containing entries in a format similar to the conversion file format (described in Section 22.1.3). Namely, an entry consists of one or more lines grouped together, with each line containing one or more "*parameter=value;*" clauses. Every line except the last must end with a semicolon. Entries are terminated by either a line that does not end in a semicolon, one or more blank lines, or both.

Each circuit check entry must specify a `NAME` parameter, providing a name or "handle" for a particular circuit, a `DESTINATION` parameter, specifying the loopback address used for this circuit, and a `RECURRENCE` parameter, specifying a recurrence rule describing how often to send messages on this circuit. Additional parameters may also be included, to specify things like the size and priority of messages to send on this circuit, or to establish thresholds relating to circuit completion timings or circuit completion failures and commands to execute upon such threshholds being exceeded.

Each time the circuit check facility executes (each time it “wakes up”, as controlled by its INTERVAL option), it will send out new messages, plus take any special actions regarding completion or lack thereof of previous circuits, according to the entries in its configuration file.

For instance, the circuit check configuration file for a sample site domain.com shown in Example 31–1 establishes two circuits, one named “domain-alpha” that loops through a system alpha.domain.com, and another named “domain-beta” that loops through another system beta.domain.com. The “domain-alpha” circuit sends out a new message every five minutes; the “domain-beta” circuit sends out a new message every 10 minutes.

Example 31–1 Sample Circuit Check Configuration File

```
name=domain-alpha;
  destination="circuitcheck%domain.com@alpha.domain.com";
  recurrence=MI5;

name=domain-beta;
  destination="circuitcheck%domain.com@beta.domain.com";
  recurrence=MI10;
```

Due to the parser used, circuit check configuration file parameter values must conform to `MIME` conventions for `Content-type:` header line parameters. In particular, this means that destination values (since they contain an @ character) must be quoted.

31.3.1.1.1 Available Circuit Check Parameters

The circuit check parameters currently available are summarized in Table 31–2 and then described individually in more detail below.

Table 31–2 Available Circuit Check Parameters

Parameter	Meaning
	Required parameters
NAME	Name (handle) for this circuit
DESTINATION	Destination address for this circuit’s messages
RECURRENCE	A recurrence rule specifying how often to send messages on this circuit

Table 31–2 (Cont.) Available Circuit Check Parameters

Parameter	Meaning
	Optional parameters
AVERAGE_THRESHOLD	A threshold value for the average circuit completion time
AVERAGE_THRESHOLD_ COMMAND	Command to execute when the AVERAGE_THRESHOLD value is exceeded
OBSOLETE_COMMAND	Command to execute when obsolete circuit message files are seen
EXPIRY	An ISO 8601 P specification of a time after which to consider messages expired
EXPIRY_COMMAND	Command to execute when an expired message is received
FAILED_COMMAND	Command to execute when messages fail
MAXIMUM_THRESHHOLD	A threshold value for the maximum circuit completion time
MAXIMUM_THRESHHOLD_ COMMAND	Command to execute when the MAXIMUM_THRESHOLD value is exceeded
OUTSTANDING_COMMAND	Command to execute when the OUTSTANDING_MAX value is exceeded
OUTSTANDING_MAX	The maximum number of outstanding messages to permit; when there are more than this number of outstanding messages, then additional new circuit messages will not be generated and sent
PRIORITY	Priority of generated messages
SIZE	Size of generated messages

AVERAGE_THRESHOLD (real number)
This parameter takes a value in seconds. If the average circuit completion time for messages in this circuit exceeds this value, PMDF will execute the command specified by the AVERAGE_THRESHOLD_COMMAND parameter value. Note that once a AVERAGE_THRESHOLD value has been exceeded, the average completion time must drop back down below AVERAGE_THRESHOLD before another execution of AVERAGE_THRESHOLD_COMMAND will be performed. That is, the message circuit check facility performs the AVERAGE_THRESHOLD_COMMAND upon AVERAGE_THRESHOLD first being exceeded—or upon any subsequent occasion when, after having been below AVERAGE_THRESHOLD, the average time again crosses over AVERAGE_THRESHOLD. During a sustained interval of being over AVERAGE_THRESHOLD, the AVERAGE_THRESHOLD_COMMAND will not be repeatedly executed.

AVERAGE_THRESHOLD_COMMAND (shell command)
This option specifies a command (a DCL command on OpenVMS or shell command on UNIX or NT) to be executed if the average circuit completion time for messages in the defined circuit exceeds the circuit's AVERAGE_THRESHOLD value.

DESTINATION (address within quotes)
The DESTINATION parameter is required in each entry. Its value must be an address, and more specifically should consist of the circuit check's own loopback address embedded within explicit routing to route the message through some channels or remote system(s). (Due to the presence of the @ character, the address must be enclosed in quotes.)

EXPIRY (ISO 8601 P time period)

The EXPIRY option may be used to specify a time after which a returning message (completing a circuit) should be ignored. ISO 8601 P format is, *e.g.*,

P*year*Y*month*M*week*W*day*DT*hour*H*minute*M*second*S

where the values *year*, *month*, *etc.*, are integer values specifying an offset (delta) from the current time. The initial P is required; other fields may be omitted, though the T is required if any time values are specified.

EXPIRY_COMMAND (shell command)

This option specifies a command (a DCL command on OpenVMS or shell command on UNIX or NT) to be executed if a message returns in greater than the EXPIRY parameter time value. An EXPIRY parameter value is often set on the assumption that any messages that are older than the specified time should be assumed to be lost and will never return; in such a case, a site may want to be notified or have other special action taken if an expired message does, in fact, return.

FAILED_COMMAND (shell command)

This option specifies a command (a DCL command on OpenVMS or shell command on UNIX or NT) to be executed if a message returns to the message circuit check facility due to being bounced while on its circuit, rather than returning normally by completing its circuit.

MAXIMUM_THRESHOLD (integer)

This parameter takes a value in seconds. If the circuit completion time for messages in this circuit exceeds this value, PMDF will execute the command specified by the MAXIMUM_THRESHOLD_COMMAND parameter value. Note that once a MAXIMUM_THRESHOLD value has been exceeded, the completion time must drop back down below MAXIMUM_THRESHOLD before another execution of MAXIMUM_THRESHOLD_COMMAND will be performed. That is, the message circuit check facility performs MAXIMUM_THRESHOLD_COMMAND upon MAXIMUM_THRESHOLD first being exceeded—or upon any subsequent occasion when, after having been below MAXIMUM_THRESHOLD, the time again crosses over MAXIMUM_THRESHOLD. During a sustained interval of being over MAXIMUM_THRESHOLD, the MAXIMUM_THRESHOLD_COMMAND will not be repeatedly executed.

MAXIMUM_THRESHOLD_COMMAND (shell command)

This option specifies a command (a DCL command on OpenVMS or shell command on UNIX or NT) to be executed if the circuit completion time for messages in the defined circuit exceeds the circuit's MAXIMUM_THRESHOLD value.

NAME (string)

The NAME parameter is required in each entry. Its value must be a string. It should be a descriptive name describing the circuit, as this is the name that, for instance, will be displayed by the PMDF CIRCUIT_CHECK/SHOW (OpenVMS) or pmdf circuit_check -show (UNIX or NT) utility.

OBSOLETE_COMMAND (shell command)

This option specifies a command (a DCL command on OpenVMS or a shell command on UNIX or NT) to be executed if obsolete message files are seen in the queue for the circuit check facility. See Section 31.3.3 below for a discussion of obsolete circuit check message files.

OUTSTANDING_COMMAND (shell command)
This option specifies a command (a DCL command on OpenVMS or a shell command on UNIX or NT) to be executed if the message circuit check facility is awaiting the return of more than `OUTSTANDING_MAX` messages.

OUTSTANDING_MAX (integer)
If more than the specified number of messages are outstanding, then the message circuit check facility will not send out any more messages until the number of outstanding messages drops below this value.

PRIORITY (string)
This option may be used to specify the Priority: of message to send on this circuit, *e.g.*, Urgent, Normal, Non-urgent.

RECURRENCE (versit vCalendar recurrence specification)
The RECURRENCE parameter is required in each entry. Its value must be a vCalendar recurrence string. This option specifies how often the entry should be executed; *i.e.*, how often the sort of message defined in the entry should be sent. The general form for a recurrence value for this option is

Uj

where *j* is an integer and *U* is a code letter (or two letters) specifying the units: `Y` for year, `M` for month, `W` for week, `D` for day, `H` for hour, or `MI` for minutes. For instance, `H1` means to recur every hour from this moment forward; `MI45` means to recur every 45 minutes from this moment forward.

Note that there is no point in using a recurrence value smaller than the circuit check facility's `INTERVAL` option.

SIZE (integer)
This option may be used to specify the size of message to send on this circuit. By default, messages of size 0 are sent; that is, messages with just headers and no message body.

31.3.2 Controlling the Circuit Check Facility

Once configured, the circuit check facility must be started up with the `PMDF STARTUP CIRCUIT_CHECK` (OpenVMS) or `pmdf startup circuit_check` (UNIX or NT) command. On OpenVMS, such a command should be inserted into your system startup procedure if you want the circuit check facility to begin running automatically after system reboots.

To restart the circuit check facility after configuration changes, use the `PMDF RESTART CIRCUIT_CHECK` (OpenVMS) or `pmdf restart circuit_check` (UNIX or NT) command.

To shut down the circuit check facility, use the `PMDF SHUTDOWN CIRCUIT_CHECK` (OpenVMS) or `pmdf shutdown circuit_check` (UNIX or NT) command.

31.3.3 Interpreting the Circuit Check Counters

The PMDF CIRCUIT_CHECK/SHOW (OpenVMS) or `pmdf circuit_check -show` (UNIX or NT) utility may be used to display the current circuit check counters. These counters are stored in an on-disk database, `circuitcheck_results_`*`nodename`*`.dat` (OpenVMS) or `circuitcheck_results.*` (UNIX or NT), located in the PMDF table directory. This database is cumulative and persists across restarts of the circuit check facility; you may delete the database if you want to clear the circuit check counters.

The counters track a number of variables.

- *Sent messages*. This is the total number of messages sent out on this circuit since the circuit check counters were last cleared (*i.e.*, since any old circuit check database was deleted and a new circuit check database created).
- *Outstanding messages*. This is the number of sent messages minus the number of completed or expired messages. It is common to configure the message circuit check facility to stop sending additional circuit check messages whenever the number of outstanding messages exceeds a specified value. That is, when the e-mail system appears to be having some delivery problem, as evidenced by a large number of message circuit check messages that have not returned since the last restart of the circuit check process, do not add further to the load by sending additional message circuit check messages.
- *Completed messages*. Besides counting the total number of messages that have completed the circuit (arrived back at the circuit check facility), the message circuit check facility also keeps track of how long the messages took to complete their circuits. Binned counts of completion time are maintained; the `CIRCUITCHECK_COMPLETED_BINS PMDF` option, Section 7.3.6, controls the bin sizes. Note that since the message check facility only wakes up periodically to check for the arrival of completed messages, the actual delivery completion times will typically be a bit less than the reported values—the reported values are intended to indicate trends rather than be precise accountings of the completion time for an individual message.
- *Minimum/average/maximum completion time values*. The message circuit check facility keeps a running average of the time for message circuit completion for messages on each circuit, as well as tracking the minimum and maximum (less than `EXPIRY`) times seen for completion of the circuit. Note that since the message check facility only wakes up periodically to check for the arrival of completed messages, the actual delivery completion times will typically be a bit less than the reported values—the reported values are intended to indicate trends rather than be precise accountings of the completion time for an individual message.
- *Expired messages*. The message circuit check facility may be configured to consider messages to *expire* if they do not return within a specified amount of time. That is, if a message has not returned from a local system within, say, a day, one might want to consider the message effectively lost—perhaps the message was deleted manually or in some other way abnormally removed from the mail system. Specifying an expiry time ensures that messages that are lost, or messages that take an abnormally long time to return due to special factors, *e.g.*, manual sidelining on a remote system, do not unduly influence the reported average and maximum completion times for messages that complete a circuit "normally".
- *Failed messages*. When a circuit check message is bounced at some point on the circuit, rather than returning normally, it is accounted as a *failed* message.

- *Obsolete message files*. When the message circuit check facility is restarted after having been previously running, there may be files corresponding to as yet unprocessed messages that have completed a circuit waiting in the queue for processing by the circuit check facility. When the circuit check facility is restarted, such message files—those generated by a previous instantiation of the circuit check facility—become *obsolete*. The "obsolete" row in the `PMDF CIRCUIT_CHECK/SHOW` (OpenVMS) or `pmdf circuit_check -show` (UNIX or NT) output refers to such left-over message files.
- *Abandoned message files*.

Example 31–2 shows a sample of `PMDF CIRCUIT_CHECK/SHOW` output on an OpenVMS system domain.com. The domain.com site is assumed to have a circuit check configuration file with five circuits defined, one through another internal domain.com system beta.domain.com ❶, one that goes through two additional internal systems gamma.domain.com and then delta.domain.com ❷, one through Message Router ❸, one through a remote X.400 MTA ❹, and one through Lotus Notes via a PMDF-LAN LN channel ❺.

Example 31–2 Sample of PMDF CIRCUIT_CHECK/SHOW Output

```
$ PMDF CIRCUIT_CHECK/SHOW
                                      2     5    15    30    60   120   240   480
beta (circuitcheck%domain.com@beta.domain.com) ❶
 Sent         2193
 Completed    2193                  294  1864    18     9     6           2
 Min/Ave/Max  1.03333/2.32877/148.367
 Obsolete     1
gamma-delta (circuitcheck%domain.com%delta.domain.com@gamma.domain.com) ❷
 Sent         2170
 Completed    2170                 2127     6    19     8     4     4     2
 Min/Ave/Max  1.01667/1.60402/127.483
 Obsolete     1
mr (circuitcheck%domain.com%PMDF@mr.domain.com) ❸
 Sent         1813
 Completed    1813                 1784     5     6     8     9           1
 Min/Ave/Max  1.03333/1.48306/127.483
 Obsolete     1
x400 (/C=US/ADMD=TELCO/PRMD=DOMAIN/S=circuitcheck/@x400.domain.com) ❹
 Sent         304
 Completed    304                   298           2     2     1           1
 Min/Ave/Max  1.03333/1.76897/127.483
lotus-notes (circuitcheck%domain.com%PMDF@lnotes.domain.com) ❺
 Sent         202
 Completed    168                   145           3                 4     6
 Min/Ave/Max  1.03333/80.6514/1.41568E3
 Obsolete     36
 Abandoned    34
```

31.3.4 Loopback Addresses for Circuit Checking

In order to set up a circuit check message path, you must specify an address that will cause a message to be routed in a loop. This may be done by sending to a remote address that is configured to then forward back to the circuit check address. Or a method that may be more convenient in some cases is to specify a form of the circuit check facility's own address that embeds the circuit check address within some explicit routing components, where the explicit routing components will cause routing along the desired path. For instance:

- When routing through an SMTP host (that allows relaying—note that many SMTP hosts will *not* allow general relaying), a %-hack address form may be used, as in ❶ or ❷ in Example 31–2 above.
- To send into and back out of Message Router, you may send to the circuit check address at the PMDF MRMAN mailbox at the PMDF-MR domain name, as in ❸ in Example 31–2 above.
- To send into and back out of ALL-IN-1 or MailWorks when connected to by PMDF-MR as an MR TS replacement, you should set up an ALL-IN-1 or MailWorks account that is configured to auto-forward back to the circuit check facility and then send to that special account.
- To send a message to an X.400 MTA that the X.400 MTA should then immediately send back to PMDF and the circuit check facility, send to PMDF-X400's own X.400 ORname stem, plus /S=circuitcheck, at the PMDF-X400 pseudodomain name, as in ❹ in Example 31–2 above.
- To loop a message through a PC-LAN mailer, use explicit routing to specify the PMDF system domain name within the PC mailer name space.
  - With cc:Mail, specify

circuitcheckaddress%*PMDFcc:mailpostoffice*@*cc_local-domain*

E.g., circuitcheck%domain.com%PMDF@ccmail.domain.com.

  - With Lotus Notes, specify

circuitcheckaddress%*PMDFlotusdomain*@*ln_local-domain*

E.g., circuitcheck%domain.com@PMDF@lnotes.domon.com.

  - With Microsoft Mail, specify

PMDFdomain/*PMDFpo*/circuitcheck@*ff_local-domain*

  - With GroupWise, specify

PMDFwpodomain.*PMDFwpopo*.*circuitcheckaddress*@*wpo_local-domain*

  - With Novell MHS, specify

circuitcheck%*PMDFmhsworkgroup*@*mhs_local-domain*

31.4 Channel Statistics Counters

PMDF has facilities to collect and monitor channel counters based upon the Mail Monitoring MIB, RFC 1566. These counters tabulate on a per channel basis the twelve items described in Table 31–3.

Table 31–3 Channel Counters

Field name	Description
RECEIVED_MESSAGES	The number of messages enqueued to the channel
SUBMITTED_MESSAGES	The number of messages enqueued by the channel
STORED_MESSAGES	The total number of messages currently stored for the channel
DELIVERED_MESSAGES	The number of messages dequeued by the channel
RECEIVED_VOLUME	The volume of messages enqueued to the channel as measured in PMDF blocks
SUBMITTED_VOLUME	The volume of messages enqueued by the channel as measured in PMDF blocks
STORED_VOLUME	The volume of messages currently stored for the channel as measured in PMDF blocks
DELIVERED_VOLUME	The volume of messages dequeued by the channel as measured in PMDF blocks
RECEIVED_RECIPIENTS	The total number of recipients specified in all messages enqueued to the channel
SUBMITTED_RECIPIENTS	The total number of recipients specified in all messages enqueued by the channel
STORED_RECIPIENTS	The total number of recipients specified in all messages currently stored for the channel
DELIVERED_RECIPIENTS	The total number of recipients specified in all messages dequeued by the channel

A PMDF block is, by default, 1024 bytes. However, this size may vary from system to system. The size of a PMDF block is controlled with the `BLOCK_SIZE` PMDF option.

It is important to note that these counters generally need to be looked at over time noting the minimum values seen. The minimums may actually be negative for some channels. Such a negative value merely means that there were messages queued for a channel at the time that its counters were zeroed (*e.g.,* the cluster-wide database of counters created). When those messages were dequeued, the associated counters for the channel were decremented therefore leading to a negative minimum. For such a counter, the correct "absolute" value is the current value less the minimum value that counter has ever held since being initialized.

31.4.1 Purpose and Use of Counters

PMDF channel counters are intended for indicating the *trend and health* of your e-mail system. PMDF channel counters are not designed nor intended to provide an accurate accounting of message traffic; for precise accounting, instead see PMDF logging as discussed above in Section 31.1. The lack of accuracy in PMDF's channel counters in an inherent aspect of their design; it is not a bug. Specifically, PMDF's channel counters adhere to what Marshall Rose calls the *fundamental axiom of management*, which is

that management must itself not interfere with proper system and network operation by consuming anything but the tiniest amount of resource.

Therefore PMDF's channel counters are implemented using the lightest weight mechanisms available, namely a shared memory section on each system (that is periodically synchronized to a disk database on OpenVMS). Channel counters do not *try harder*: if an attempt to map the section fails, no information is recorded; if one of the locks in the section cannot be obtained almost immediately, no information is recorded; when a system is shut down, the information contained in the in-memory section is lost forever. Section 31.4.2 and Section 31.4.3 provide further discussion of the implementation of counters.

31.4.1.1 Example of Counters Interpretation

Example 31–3 shows a sample excerpt of counters data, as might be seen using the PMDF COUNTERS/SHOW (OpenVMS) or `pmdf counters -show` (UNIX and NT) utility.

Example 31–3 Sample of Counters Data

```
Channel                    Messages  Recipients      Blocks
-----------------------  ----------  ----------  ----------
directory
    Received                   6523        9042       69694  ❶
    Stored                        4           4         149  ❷
    Delivered                  6519        9038       69545  (6500 first time) ❸
    Submitted                  6811        9019       71123  ❹
    Attempted                    21          25         287  ❺
    Rejected                      0           0           0  ❻
    Failed                        0           0           0  ❼

    Queue time/count        100020/6519 = 15.34  ❽
    Queue first time/count  31525/6500 = 4.85    ❾
```

In this example:

❶ The "Received" value represents a count of messages coming from any channel to the channel named directory; that is, messages enqueued ("E" records in the `mail.log*` file) to the directory channel by any other channel.

❷ The "Stored" value represents a count of messages stored in the channel queue to be delivered. This will generally correspond to the number of entries currently stored for the channel in the PMDF queue cache database.

❸ The "Delivered" value represents a count of messages which have been processed (dequeued) by the channel directory, *i.e.*, "D" records in the `mail.log*` file. A dequeue operation may either correspond to a successful "delivery" (that is, an enqueue to another channel), or to a dequeue due to the message being returned to the sender. This will generally correspond to the number Received minus the number Stored. PMDF also keeps track of how many of the messages dequeued were dequeued upon first attempt; this number is shown in parentheses.

❹ The "Submitted" value represents a count of messages which have been enqueued ("E" records in the `mail.log*` file) from the channel directory to any other channel.

❺ The "Attempted" value represents a count of messages which have experienced temporary problems in dequeuing, *i.e.*, "Q" or "Z" records in the `mail.log*` file.

❻ The "Rejected" value represents a count of attempted enqueues to the channel which have been rejected, *i.e.*, "J" records in the `mail.log*` file.

❼ The "Failed" value represents a count of attempted dequeues which have failed, *i.e.*, "R" records in the `mail.log*` file.

❽ The "Queue time/count" represents the average time-spent-in-queue for the delivered messages. This includes both the messages delivered upon the first attempt—see ❾—and the messages that required additional delivery attempts (hence typically spent noticeable time waiting fallow in the queue).

❾ The "Queue first time/count" represents the average time-spent-in-queue for the messages delivered upon the first attempt.

Note that in this example, the number of messages Submitted is greater than the number delivered. This is often the case, since each message the channel dequeues (delivers) will result in at least one new message enqueued (submitted) but possibly more than one. For example, if a message has two recipients reached via different channels, then two enqueues will be required. Or if a message bounces, a copy will go back to the sender and another copy may be sent to the postmaster. Usually that will be two submissions (unless both are reached through the same channel).

When interpreting counters values, keep in mind the discussion of Section 31.4.1: counters are neither intended nor expected to be to-the-message accurate. Rather, counters are intended to give a general idea of current message traffic trends, while causing as little impact as possible on actual operation; and counter information will be discarded rather than recorded whenever recording it would be burdensome for operation.

31.4.2 Implementation on OpenVMS

For performance reasons, each node running PMDF keeps a cache of channel counters in memory using a permanent, global, writeable page-file section. As a process on a node enqueues and dequeues messages, it updates the counters in its own in-memory cache. The DCL command `PMDF COUNTERS/SYNCHRONIZE` or the PMDF QM command `COUNTERS SYNCHRONIZE` may be used to cause each node in the cluster to merge its node specific, in-memory cache with the cluster-wide, on-disk database of channel counters, `PMDF_TABLE:counters.dat`. The synchronization is accomplished through a combination of light-weight detached process running on each node and cluster-wide resource locks. The synchronization command signals each detached process on each node. Upon being signalled, each detached process adds the values of its in-memory counters to those in the cluster-wide, on-disk database and then zeroes its own in-memory counters if the update was successful. SYSLCK privilege is required to perform a synchronization.

The DCL command PMDF COUNTERS/SHOW or the PMDF QM command COUNTERS SHOW may be used to show the values of the cluster-wide counters as stored in the on-disk database. Note that these commands will automatically perform a synchronization of the node-specific caches with the cluster-wide database. The PMDF counters may also be viewed via a web interface; see Section 31.7.

The DCL command PMDF COUNTERS/CLEAR or the PMDF QM command COUNTERS CLEAR may be used to reset the counters to zero.

When the command procedure SYS$STARTUP:pmdf_startup.com is executed, it starts running the single detached process used to perform synchronizations for that node. The process, upon starting, will create the global section representing the in-memory cache and ensure that the cluster-wide database of channel counters exists.[2] After performing these two steps it will trim its working set and hibernate, waiting for synchronization commands. SYSGBL and PRMGLB privileges are required to create the global section. To prevent the detached processes from being started, define the logical PMDF_NOCOUNTERS prior to executing pmdf_startup.com.

By default, the counters synchronization process automatically synchronizes the in-memory cache values to the on-disk database every thirty minutes. This time interval may be changed by defining a PMDF_COUNTER_INTERVAL logical. If defined, PMDF_COUNTER_INTERVAL should be a system-level logical definition, equating to an OpenVMS delta time, *e.g.*,

```
$ DEFINE/SYSTEM PMDF_COUNTER_INTERVAL "0 00:15:00"
```

Note that SYS$BINTIM requires an initial day value; be sure to specify the leading 0 for the day value.

The PMDF RESTART and SHUTDOWN commands may be used to restart or shutdown the detached process on all nodes. However, there should be no need to do this. Note that the processes are not affected by changes to the PMDF configuration.

31.4.3 Implementation on UNIX and NT

For performance reasons, a node running PMDF keeps a cache of channel counters in memory using a shared memory section (UNIX) or shared file-mapping object (NT). As processes on the node enqueue and dequeue messages, they update the counters in this in-memory cache. If the in-memory section does not exist when a channel runs, the section will be created automatically. (The pmdf startup command also creates the in-memory section, if it does not exist.)

The command pmdf counters -show or the pmdf qm command counters show may be used to show the values of the counters. The PMDF counters may also be viewed via a web interface; see Section 31.7.

The command pmdf counters -clear or the pmdf qm command counters clear may be used to reset the counters to zero.

[2] On OpenVMS systems a $MGBLSC call is used to create a system-wide, writeable global section. That section is then turned into a permanent, writeable, global page-file section via a call to $CRMPSC.

31.5 HP Commander scanning module (OpenVMS and Tru64 UNIX only)

Sites running the HP-supplied HP Commander monitoring package (also known as Enterprise Mail Monitor, EMM, or PolyCenter MAILbus Monitor (PMM)), can monitor PMDF using the PMM scanning module supplied with PMDF. The scanning module sends to PMM software version, time zone, queue, job, and monitor records. The contents of each of these records is described in Section 31.5.3. Note that PMDF channel counters are sent in the monitor records.

31.5.1 Required Software

In order to use the PMDF PMM scanning module, sites must be running HP's Polycenter MAILbus Monitor V2.0 or MAILbus Monitor V2.1 for OpenVMS VAX, or OpenVMS Alpha.

31.5.2 Configuration

To configure PMM to scan PMDF, it is necessary to add an entry for the PMDF scanning module to the PMM `namon$server.ini` file. The entry on OpenVMS should be

```
[Entity PMDF]
Command="RUN PMDF_EXE:pmm_scanner.exe"
FullName=PMDF e-mail Interconnect
MaxRun=1200
```

Once that entry has been made, restart the NAMON$SERVER process so that the change to the file will be seen; *e.g.*, on OpenVMS

```
$ @SYS$STARTUP:namon$shutdown.com
$ @SYS$STARTUP:namon$startup.com
```

After the server has restarted, you should then be able to see PMDF appear as one of the scannable services in the PMM workstation display.

31.5.3 Operation

Each time PMM scans PMDF, it activates the PMDF scanning module. This module initializes PMDF. If PMDF cannot be initialized, either because PMDF has not been started or because of a serious configuration error, a fatal error is signalled back to PMM. If PMDF is running, the module "scans" PMDF and transmits back to PMM five different types of scanning records:

Software version

The software version record gives the version and link time of the PMDF shareable image, `PMDF_SHARE_LIBRARY` (OpenVMS) or `libpmdf.so` (UNIX). This is the single most critical image in PMDF and contains the core PMDF library routines. If this information cannot be obtained, an error is signalled to PMM.

Time zone

The time zone records show the numerical offset from Greenwich mean time (GMT) as well as the mnemonic time zone code (*e.g.,* EST for Eastern Standard Time). If this information cannot be obtained, an error is signalled to PMM.

Queue (OpenVMS only)

A queue record containing current queue status information is sent for the `MAIL$BATCH` queue as well as each queue cited in the PMDF configuration file. If a queue cannot be located or information about it obtained, an error is signalled to PMM. If the queue is stalled, stopping, or stopped, a warning is signalled to PMM.

Job (OpenVMS only)

For each job queued to a PMDF processing queue, a job record is sent to PMM. Errors will be signalled if the periodic delivery or message bouncer jobs are missing from the `MAIL$BATCH` queue.

NCL

The PMDF channel counters for each channel are sent in twelve records; each record containing the value of a particular counter in its "Counter Value" field. For each set of twelve records, the name of the channel is given in the "Object Name" field. Each of the twelve records is distinguished by the value of the "Identifier" field in the record. The different values of that field and their interpretation are given in the table below.

	Identifier	Description
†	Received messages	Cumulative count of the messages enqueued to the channel
†	Enqueued messages	Cumulative count of the messages enqueued (sent) by the channel
†	Stored messages	Current count of the messages held by the channel and waiting to be processed
†	Dequeued messages	Cumulative count of the messages processed by the channel
	Received recipients	Cumulative count of the message recipients enqeued to the channel
	Enqueued recipients	Cumulative count of the message recipients enqueued (sent) by the channel
	Stored recipients	Current count of the message recipients held by the channel and waiting to be processed
	Dequeued recipients	Cumulative count of the message recipients processed by the channel
	Received volume	Cumulative size of the messages enqueued to the channel
	Enqueued volume	Cumulative size of the messages enqueued (sent) by the channel
	Stored volume	Current size of the messages held by the channel and awaiting processing
	Dequeued volume	Current size of the messages processed by the channel

†For these counters, the rate of change (message/hour) and tendency (messages/hour/hour) are also reported, respectively, in the "Counter Derivative" and "Counter Tendency" fields.

If the channel counters cannot be obtained, an error is signalled to PMM.

Once all of these records have been transmitted back to PMM, the scanning module exits, awaiting a subsequent invocation by PMM.

31.6 SNMP Support on OpenVMS

SNMP subagents are available to serve out the PMDF channel counters using the Mail and Directory Management (MADMAN) SNMP MIB described in RFCs 1565 and 1566. Presently, SNMP subagents are available for use with Process Software TCPware V5.1 and later.

An SNMP subagent for TCPware is available from Process Software. Directions for configuring TCPware to use the subagent are provided in Section 31.6.3. Note that you must be running TCPware V5.1-4 or later. If you are running V5.1-4, then you will also need to apply a patch kit to the TCPware SNMP agent. The kit is available via FTP from ftp.process.com as a `BACKUP` saveset. For OpenVMS Alpha, it is the file `/support/51_4/snmpd_v514a_axp.inc`; for OpenVMS VAX it is the file `/support/51_4/snmpd_v514a_vax.inc`.

31.6.1 Operation

In order for the SNMP subagents to properly operate, the PMDF detached counters processes must be running on each cluster node with PMDF. Those detached processes are automatically started at system boot time by the PMDF startup procedure, `SYS$STARTUP:pmdf_startup.com`. They may be restarted with the command

```
$ PMDF RESTART COUNTERS
```

and shutdown with the command

```
$ PMDF SHUTDOWN COUNTERS
```

Note that the PMDF RESTART command will only restart an already running detached counters process; it will not start a detached process when none are running. To start a detached process, issue the command

```
$ @PMDF_COM:start_synch_counters.com
```

31.6.2 MIB Variables Served

The subagents serve out selected variables from the MADMAN MIBs.[3] Specifically, those variables from the applicationTable, mtaTable, and mtaGroupTable tables as shown in Table 31–4.

[3] See RFCs 1565 and 1566 for the specification of those MIBs. Copies of those RFCs may be found in the directory `pmdf_root:[doc.rfc]`.

Table 31–4 Supported MIB Variables

applicationTable variables		
Variable name	**OID**	**Syntax**
applName	mib-2.27.1.1.2	String
applVersion	mib-2.27.1.1.4	String

mtaTable variables		
Variable name	**OID**	**Syntax**
mtaReceivedMessages	mib-2.28.1.1.1	Counter32
mtaStoredMessages	mib-2.28.1.1.2	Gauge32
mtaTransmittedMessages	mib-2.28.1.1.3	Counter32
mtaReceivedVolume	mib-2.28.1.1.4	Counter32
mtaStoredVolume	mib-2.28.1.1.5	Gauge32
mtaTransmittedVolume	mib-2.28.1.1.6	Counter32
mtaReceivedRecipients	mib-2.28.1.1.7	Counter32
mtaStoredRecipients	mib-2.28.1.1.8	Gauge32
mtaTransmittedRecipients	mib-2.28.1.1.9	Counter32

mtaGroupTable variables		
Variable name	**OID**	**Syntax**
mtaGroupReceivedMessages	mib-2.28.2.1.2	Counter32
mtaGroupStoredMessages	mib-2.28.2.1.4	Gauge32
mtaGroupTransmittedMessages	mib-2.28.2.1.5	Counter32
mtaGroupReceivedVolume	mib-2.28.2.1.6	Counter32
mtaGroupStoredVolume	mib-2.28.2.1.7	Gauge32
mtaGroupTransmittedVolume	mib-2.28.2.1.8	Counter32
mtaGroupReceivedRecipients	mib-2.28.2.1.9	Counter32
mtaGroupStoredRecipients	mib-2.28.2.1.10	Gauge32
mtaGroupTransmittedRecipients	mib-2.28.2.1.11	Counter32
mtaGroupName	mib-2.28.2.1.25	String

Note: the OID for mib-2 is 1.3.6.1.2.1.

Each PMDF channel is identified with with an MTA group. Thus, for each channel, there will be a row in the mtaGroupTable. For example, if there are M channels, the OID mib-2.28.2.1.25.n gives the name of the channel associated with the nth row in the table where n satisfies $1 \leq n \leq M$.

Only one application and MTA is recognized by the subagent and consequently there is only one row in the applicationTable and mtaTable tables. The only valid instance identifier for those two tables is thus ".1"; *i.e.,* for either table, the OID for an instance of a variable is formed by taking the OID of the variable and appending ".1" to it. For example, a `get` operation on mib-2.27.1.1.4.1 would return the version number of PMDF.

Each row of the mtaGroupTable table corresponds to a set of PMDF channel counters maintained by PMDF. A description of each variable is given in Table 31–5. These counters may be directly manipulated on OpenVMS PMDF systems with either the `PMDF COUNTERS` utility or the `PMDF QM/MAINTENANCE` utility. Refer to the Section 31.4 for further information on the PMDF channel counters.

Table 31–5 Variable Descriptions

mtaGroupTable variable	PMDF counter	Description
mtaGroupReceivedMessages	RECEIVED_MESSAGES	Count of messages enqueued to the channel.
mtaGroupStoredMessages	STORED_MESSAGES	Count of messages enqueued to the channel but not yet delivered.
mtaGroupTransmittedMessages	DELIVERED_MESSAGES	Count of messages delivered (dequeued) by the channel.
mtaGroupReceivedVolume	RECEIVED_VOLUME	Volume of messages enqueued to the channel as measured in Kbytes = 1024 bytes.
mtaGroupStoredVolume	STORED_VOLUME	Volume of messages enqueued to the channel but not yet delivered as measured in Kbytes.
mtaGroupTransmittedVolume	DELIVERED_VOLUME	Volume of messages which have been delivered (dequeued) by the channel as measured in Kbytes.
mtaGroupReceivedRecipients	RECEIVED_RECIPIENTS	Volume of messages enqueued to the channel as measured by the total number of envelope recipient addresses.
mtaGroupStoredRecipients	STORED_RECIPIENTS	Volume of messages enqueued to the channel but not yet delivered as measured by the total number of envelope recipient addresses.
mtaGroupTransmittedRecipients	DELIVERED_RECIPIENTS	Volume of messages which have been delivered (dequeued) by the channel as measured by the total number of envelope recipient addresses.
mtaGroupName		Name of the channel.

The values in the mtaTable correspond to the column sums of the mtaGroupTable; *e.g.,* mtaReceivedMessages is the sum over all rows of the mtaGroupTable column mtaGroupReceivedMessages.

Note: The underlying PMDF channel counters may take on negative values. However, the corresponding MIB variables must be non-negative. To reconcile this difference, the subagent tracks the minimum value seen for each channel counter and then uses that minimum to adjust the MIB variable such that it has a minimum of zero. This is done by subtracting the minimum value from the counter when that minimum is less than zero. For this reason, the values of the counters displayed with the `PMDF COUNTERS` command may differ from those displayed from an SNMP client.

31.6.3 Configuring the TCPware Subagent

In order to have TCPware's SNMP agent serve out PMDF's channel counters, you must configure TCPware to know of the PMDF SNMP subagent. This is done with the TCPware command:

```
$ @TCPWARE:cnfnet.com SNMP
```

As it runs, that procedure will ask several questions including:

1. `Do you want to activate the SNMP agent on this host [YES]:`
   Answer `YES` to this question in order to activate the TCPware SNMP agent.

2. `Do you want to configure subagent(s) on this host[NO]:`
   Answer `YES` to this question in order to enable the PMDF SNMP subagent.

3. `Enter the name of the shareable image without .EXE:`
   Answer `PMDF_EXE:PMDF_SNMP_TCPWARE` to this question. If you have additional subagents, enter their names in response to the additional prompts. When there are no other subagent names to supply, enter a blank line.

4. `Do you want to restart the SNMP Agent [NO]:`
   Answer `YES` so as to start or restart the TCPware SNMP agent.

Configuring and starting the TCPware SNMP agent completes the configuration of the PMDF SNMP subagent. All operation of the subagent is automatically handled by the TCPware SNMP agent. Once the agent is configured and restarted, you can query the MIB variables described in Table 31–4 from an SNMP client.

Operation of the subagent is handled by TCPware. Consult the TCPware SNMP Services Management chapter of the *TCPware for OpenVMS Management Guide* for details.

31.7 Web-based Counter Monitoring

The following section documents the web-based monitoring facilities available in PMDF. These facilities require both TCP/IP support and a web client in order to use.

Web-based monitoring of PMDF is accomplished through an HTTP CGI (Common Gateway Interface). This interface is reached through the PMDF HTTP server and accepts URL-encoded commands via either `HTTP GET` or `POST` requests.

Presently, the CGI only provides passive monitoring of PMDF; active management is not supported but planned for future releases. Basic PMDF information, channel counters and derived statistics, and the presence of `.HELD` message files may be monitored. Additionally, on OpenVMS systems, PMDF processing queues and jobs may be monitored.

Section 31.7.1 describes how to use the default monitoring configuration. Note that this configuration is merely an example, albeit a fairly useful one. Sites with HTML experience can change the layouts, navigation, and choices or presentation of monitored data. Sites familiar with HTML scripting languages such as Java or JavaScript can add intelligence to how their web clients analyze the monitoring data available through the CGI. See Section 31.7.3 for complete details.

31.7.1 Using the Sample Monitoring Configuration

To use the monitoring CGI, you must first configure the PMDF Service Dispatcher and PMDF HTTP server. That is done by running the Dispatcher configuration utility as described in the *PMDF Installation Guide*. If you have not configured the Dispatcher, then do so now.

Before connecting to the monitoring CGI, ensure that your web browser supports HTML tables and frames. Moreover, if you intend to use the JavaScript enhanced monitoring, also check that your browser supports JavaScript and that the JavaScript interpreter is enabled.[4]

To connect to the monitoring CGI with your web browser, open the URL

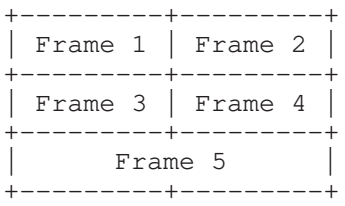
```
http://host:7633/monitor/
```

In place of `host`, use the actual IP host name of the system running the PMDF HTTP server. If you chose to run the PMDF HTTP server on a port other than port 7633, then specify that port number in place of `7633` in the above URL.

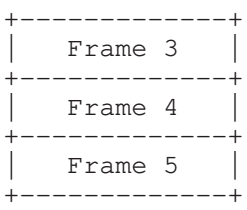
After opening the above URL, you will be presented with an initial explanatory page. From the bottom of that page, select either the JavaScript enhanced monitoring link or the normal monitoring link. Note that you may only see the normal monitoring link; if so, then your browser does not support JavaScript.

After you select the style of monitoring to perform, your browser window will split into multiple frames.

When monitoring OpenVMS PMDF platforms, five frames will appear,



When monitoring UNIX PMDF platforms, three frames will appear,



The contents of the frames are as follows:

[4] Do not confuse JavaScript with Java; they are two completely different scripting languages. Morever, if using Netscape Navigator, then you must use Navigator 3.0 or later; if using Microsoft Internet Explorer, then you must use Internet Explorer 3.0 or later.

Frame 1: All OpenVMS processing queues used by PMDF throughout the cluster.

Frame 2: When you click on a link in Frame 1, the results appear in Frame 2.

Frame 3: Channel counters for all your PMDF channels; cluster-wide on OpenVMS.

Frame 4: When you click on a link in Frame 3, the results appear in Frame 4.

Frame 5: Warning about `.HELD` messages; this frame only appears when using JavaScript enhanced monitoring.

The contents of Frames 1 through 4 will be updated once every two minutes; the contents of Frame 5 will be updated only once every thirty minutes.[5]

When JavaScript enhanced monitoring is used, the background colors of the individual frames will indicate whether or not a problem has been detected. For each frame, the background will be green when no problems are detected. When a problem is detected, the frame background will be red and the problem indicated in flashing text and called out with a burning trash can. Frame-by-frame, the problem criteria are as follows:

Frame 1: Red background if a queue is closed, disconnected, stalled, stopped, stopping, or has a stop pending, or if the periodic delivery job or message bouncer job is missing from the queues.

Frame 2: Red background if a queue is closed, disconnected, stalled, stopped, stopping, or has a stop pending.

Frame 3: Red background if the count of stored messages in a channel exceeds the threshold specified in the monitoring option file. See Section 31.7.2 for details.

Frame 4: Red background if the count of stored messages in a channel exceeds the threshold specified in the monitoring option file. See Section 31.7.2 for details.

Frame 5: Red background if `.HELD` message files are detected.

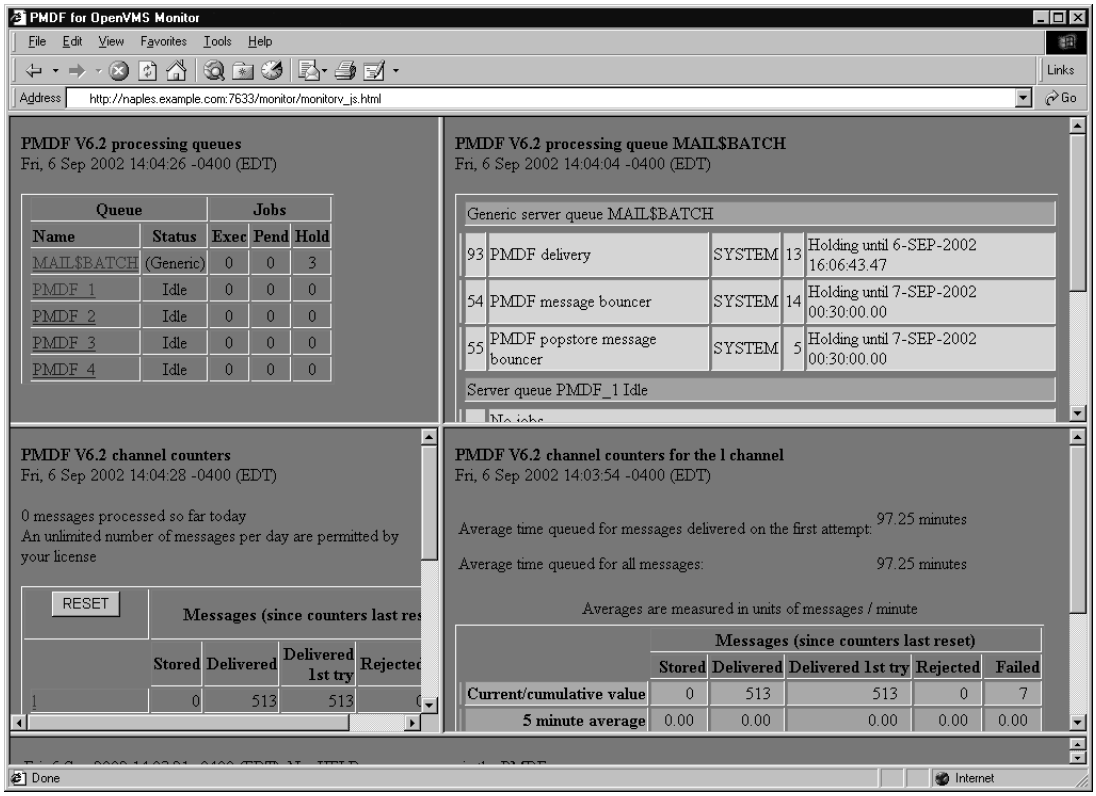
Figure 31–17 shows a sample of the web monitor display on OpenVMS.

31.7.2 The Monitoring Option File

An option file may be used to specify alarm thresholds for stored message counts. These thresholds may be used by intelligent monitoring clients to determine when a channel has suspiciously too many stored (*i.e.,* queued) messages. The sample monitoring configuration described in Section 31.7.1 uses these thresholds for the JavaScript enhanced monitoring.

[5] Owing to some Netscape Navigator 3.0 bugs, you may find that the frames are not actually updated. It's possible to get Navigator into a mode whereby it incorrectly updates frames from a client cache rather then from the remote server. For instance, resizing the frames can lead to this state. To correct this situation, clear Navigator's in-memory and on-disk caches with the Options, Network Preferences..., Cache menu. After clearing the caches, reload each frame one by one: select a frame and then select the View, Reload Frame menu item; repeat for the other frames. Do not reload the entire window as that will undo any adjustments made to the relative frame sizes.

Figure 31–17 Sample Web Monitor Display on OpenVMS



The option file is called `PMDF_TABLE:monitor_option.` (OpenVMS) or `/pmdf/table/monitor_option` (UNIX) or `C:\pmdf\table\monitor_option` (NT).

By default, the threshold level is 1500 messages; that is, the value reported for the `%counter_stored_messages_threshold` is 1500, by default. To specify a different default threshold level, in the option file specify

```
STORED_THRESHOLD=value
```

where *value* is the desired value. To set a threshold for a specific channel, specify the option

```
channel-name_STORED_THRESHOLD=value
```

where *channel-name* is the name of the channel for which to set the threshold.

The sample option file shown below, sets a default threshold of 200 messages and thresholds of 500 messages for the local and tcp_local channels:

```
STORED_THRESHOLD=200
L_STORED_THRESHOLD=500
TCP_LOCAL_STORED_THRESHOLD=500
```

31.7.3 Monitoring Customization

In order to customize the monitoring interface, it is first necessary to understand how the monitoring CGI processes HTTP requests and formulates HTTP responses. This is described in Section 31.7.3.1 and Section 31.7.3.2. Following those descriptions, Section 31.7.3.3 describes the individual commands which can be embedded in those requests.

31.7.3.1 Processing HTTP Requests

The CGI interface responds to HTTP GET and POST requests by parsing the request for a monitoring command and generating the appropriate response. The commands take the general form

```
command=command-name&parameter-name-1=parameter-value-1&
  ...&parameter-name-N=parameter-value-N
```

(In the above, the line has been wrapped for typographic reasons.) The allowed command names and associated parameters are described in Section 31.7.3.3.

In the case of GET requests, commands are embedded in the URL to retrieve as follows:

```
http://host:7633/monitor/?command=command-name&parameter-name-1=
  parameter-value-1&...&parameter-name-N=parameter-value-N
```

And in POST requests the content of the request contains the command in URL-encoded form:

```
command=command-name&parameter-name-1=
  parameter-value-1&...&parameter-name-N=parameter-value-N
```

If the command cannot be extracted from the request, an HTTP 5yz error response is sent back to the client. If the command can be extracted but cannot be parsed or successfully executed, a successful HTTP 200 response is sent back; the content of the HTTP response will be formatted as per the error formatting directions specified in the monitoring command. If those directions could not be extracted from the command, then an HTTP 500 error response is returned. See Section 31.7.3.2 for further details.

31.7.3.2 Generating HTTP Responses

After processing an HTTP request from a client, the result of processing the monitoring command is sent back as an HTTP response to the client. The format of the response is governed by formatting files specified in the command from the client. That is, the request from the client includes the names of formatting files on the CGI server system that are to be used to format the response sent back to the client. On OpenVMS systems, these files must reside in the `pmdf_root:[www.monitor]` directory; on UNIX systems, these files must reside in the `/pmdf/www/monitor/` directory or a subdirectory thereof.

The formatting files may contain text to be copied verbatim into the HTTP response as well as directives to substitute in values associated with the monitoring information collected by the CGI. There are three basic types of formatting files: success, error, and command-specific files.

After the CGI parses a request and executes it, the results of the operation are sent back to the HTTP client using the following formatting steps:

1. The command-specific formatting files are used to format the data collected by the monitoring command. The files will be consulted once for each instance of the entity to be monitored. In the case of generic PMDF information and `.HELD` message files, the formatting files are consulted only once. In the case of channel counters and processing queues, the files are consulted once for each channel or queue monitored. The command-specific formatting files are described in the sections describing the specific commands.
2. If the preceding step is successful, the content of the success formatting file is used as the response to the client. Each line of the file is copied to the content of the HTTP response. Any line beginning with `%s` is replaced with the formatted data generated in the first step. The HTTP response sent back to the client will have an HTTP 200 status code.
3. If the first step failed, the content of the error formatting file is sent as the response to the client. Each line of the file is copied to the content of the HTTP response. Any line beginning with `%s` is replaced with the output, if any, of the first step as well as any error messages. The HTTP response sent back to the client will have an HTTP 200 status code.

In the command-specific formatting files, command-specific substitution strings may appear. These strings all begin with the percent character, `%`. When such a string is encountered, the value it references is substituted into the HTTP response. For instance, the formatting file

```
%first{<TABLE>}
%first{<TR><TH>Channel<TH>Queued messages}
<TR><TD>%counter_channel<TD>%counter_stored_messages
%last{</TABLE>}
```

might be used in conjunction with the `show_counters` command to produce an HTML table of the PMDF channels and the number of messages queued to each channel; *e.g.*,

```
<TABLE>
<TR><TH>Channel<TH>Queued Messages
<TR><TD>conversion<TD>3
<TR><TD>l<TD>29
<TR><TD>tcp_local<TD>88
</TABLE>
```

In the tables describing each substitution string, the type of data associated with the substitution string is stated. These types are:

Type	Description
int	Signed integer

Type	Description
string	ASCII text string
uint	Unsigned integer

In addition, the default formatting string used to format the data is also shown in the tables. The formatting strings follow the C programming language convention for formatting strings passed to the sprintf( ) C run-time library routine. Alternate formatting strings may be used by enclosing them in braces, `{}`, and appending them to the substitution string. For instance,

```
%counter_delivered_volume{%.1f} Kbyte%s
```

might be used to limit the `%counter_delivered_volume` to a single digit of precision after the decimal point.

Five substitution strings, `%first`, `%last`, `%!first`, `%!last`, and `%none`, deserve special attention. These first four strings substitute into the output specific text when formatting, respectively, the first, the last, not the first, or not the last instance of the data to be formatted. The text to be substituted in must be enclosed in braces, `{}`, following the substitution string.[6] For example, suppose that counters for several channels are to be formatted using the following formatting file:

```
%first{<TABLE>}
%counter_channel
%last{</TABLE>}
```

In that case, when information for the first channel is formatted, the text `<TABLE>` will be output followed by the channel name (`%counter_channel`). When information for the last channel is formatted, the text `</TABLE>` will be output following the channel name.

The `%none` substitution string supplies text to output when there are no instances of the monitored data to display. For example, when there are no jobs in a queue to display.

31.7.3.3 Monitoring Commands

As described in Section 31.7.3.1, monitoring commands take the general form

```
command=command-name&parameter-name-1=parameter-value-1&
  ...&parameter-name-N=parameter-value-N
```

In the above, *command-name* gives the name of the monitoring command to execute. It is then followed by two or more parameters which provide supplemental information relevant to the operation to be performed.

[6] At present, substitution strings appearing within the text to be substituted are ignored and treated as literal text.

The valid command names are listed in the table below and described in the following subsections.

Command name	Section	Description
show_counters	31.7.3.3.1	Show channel counters
show_held	31.7.3.3.2	Determine if there are any .HELD message files
show_pmdf	31.7.3.3.3	Show generic information about the PMDF configuration
show_queue	31.7.3.3.4	Show information about OpenVMS processing queues

31.7.3.3.1 Channel Counters: command=show_counters

The show_counters command displays channel counters and derived statistics for one or more channels. Parameter names and their values accepted by the command are listed in the table below.

Parameter		Description
counter_format=*file-spec*	Required	Name of the formatting file to use to format channel counter information. The recognized substitution strings for this command are listed in Table 31–6 and Table 31–7.
channel_name=*name*	Optional	Name of the channel to display information for. Wildcards are permitted. If this option is not specified, * is assumed and information for all channels will be displayed.
success_format=*file-spec*	Required	Name of the formatting file to use to format the results when the command is successful.
error_format=*file-spec*	Required	Name of the formatting file to use to format the results when the command fails.

Each of the *file-spec* file specifications must be relative file paths specifying files in the pmdf_root:[www.monitor] directory (OpenVMS) or the /pmdf/www/monitor/ directory (UNIX).

For each channel matching the channel name pattern, the counter format file will be used to format the information for that channel. If the command is successful, the results are then formatted as per the success format file; otherwise, the error format file will be used to generate an error response.

An example URL for a GET command might be

```
http://host:7633/monitor/?command=show_counters&channel_name=tcp*&
  counter_format=counters.txt&success_format=csuccess.txt&
  error_format=cerror.txt
```

Table 31–6 General Substitution Strings

Substitution string	Type	Format	Description
%last	string	%s	Text string to display if this is the last formatting pass.
%none	string	%s	Text string to display if there is no information to format.
%first	string	%s	Text string to display if this is the first formatting pass.
%!first	string	%s	Text string to display if this is not the first formatting pass.
%!last	string	%s	Text string to display if this is not the last formatting pass.
%S	char	%s	Output an S if the previously displayed numeric value had a non-singular value.
%s	char	%s	Output an s if the previously displayed numeric value had a non-singular value.
%host	string	%s	Display the TCP/IP host name of the node on which the monitoring data was collected.
%image_ident	string	%s	PMDF version number as recorded in the PMDF shared library.
%image_link_date	string	%s	Date and time the PMDF shared library was linked.
%node	string	%s	Display the DECnet node name (VMS) or TCP/IP host name (UNIX) of the node on which the monitoring data was collected.
%time	string	%s	Display the date and time at which the monitoring data was collected.
%time_zone_name	string	%s	Display the name for the time zone under which the monitoring data was collected.
%time_zone_offset	int	%+04d	Display the numeric time zone offset, in minutes, for the time zone under which the monitoring data was collected.
%time_zone_offset_hm	int	%s	Display the numeric time zone offset, in hhmm format, for the time zone under which the monitoring data was collected.

Table 31–7 Substitution Strings for Use with the show_counters Command

Substitution string	Type	Format	Description
%counter_channel	string	%s	The channel's name.
%counter_delivered_messages	int	%d	Total cumulative count of messages processed (dequeued) by the channel.

Table 31–7 (Cont.) Substitution Strings for Use with the `show_counters` Command

Substitution string	Type	Format	Description
%counter_delivered_messages_rate_5 %counter_delivered_messages_rate_5_min %counter_delivered_messages_rate_5_max %counter_delivered_messages_acc_5 %counter_delivered_messages_acc_5_min %counter_delivered_messages_acc_5_max %counter_delivered_messages_rate_15 %counter_delivered_messages_rate_15_min %counter_delivered_messages_rate_15_max %counter_delivered_messages_acc_15 %counter_delivered_messages_acc_15_min %counter_delivered_messages_acc_15_max %counter_delivered_messages_rate_60 %counter_delivered_messages_rate_60_min %counter_delivered_messages_rate_60_max %counter_delivered_messages_acc_60 %counter_delivered_messages_acc_60_min %counter_delivered_messages_acc_60_max	float	%.2f	Rate of change (_rate_) and acceleration (_acc_), measured in messages per minute and messages per minute per minute, averaged over the past 5, 15, and 60 minutes. Also, the minimum and maximum values for these values.
%counter_delivered_recipients	int	%d	Total cumulative count of message recipients processed (dequeued) by the channel.
%counter_delivered_volume_b %counter_delivered_volume_k %counter_delivered_volume_m %counter_delivered_volume_g	float	%.2f	Total cumulative count of message volume processed (dequeued) by then channel as measured in bytes, kilobytes, megabytes, and gigabytes.
%counter_received_messages	int	%d	Total cumulative count of messages sent to the channel (messages enqueued to the channel).
%counter_received_messages_rate_5 %counter_received_messages_rate_5_min %counter_received_messages_rate_5_max %counter_received_messages_acc_5 %counter_received_messages_acc_5_min %counter_received_messages_acc_5_max %counter_received_messages_rate_15 %counter_received_messages_rate_15_min %counter_received_messages_rate_15_max %counter_received_messages_acc_15 %counter_received_messages_acc_15_min %counter_received_messages_acc_15_max %counter_received_messages_rate_60 %counter_received_messages_rate_60_min %counter_received_messages_rate_60_max %counter_received_messages_acc_60 %counter_received_messages_acc_60_min %counter_received_messages_acc_60_max	float	%.2f	Rate of change (_rate_) and acceleration (_acc_), measured in messages per minute and messages per minute per minute, averaged over the past 5, 15, and 60 minutes. Also, the minimum and maximum values for these values.
%counter_received_recipients	int	%d	Total cumulative count of message recipients sent to the channel (recipients enqueued to the channel).

Table 31–7 (Cont.) Substitution Strings for Use with the `show_counters` Command

Substitution string	Type	Format	Description
%counter_received_volume_b %counter_received_volume_k %counter_received_volume_m %counter_received_volume_g	float	%.2f	Total cumulative count of message volume sent to the channel (enqueued to the channel) as measured in bytes, kilobytes, megabytes, and gigabytes.
%counter_stored_messages %counter_stored_messages_min %counter_stored_messages_max	int	%d	Count of messages currently enqueued to the channel as well as the minimum and maximum recorded counts for these values.
%counter_stored_messages_avg_5 %counter_stored_messages_avg_5_min %counter_stored_messages_avg_5_max %counter_stored_messages_avg_15 %counter_stored_messages_avg_15_min %counter_stored_messages_avg_15_max %counter_stored_messages_avg_60 %counter_stored_messages_avg_60_min %counter_stored_messages_avg_60_max	float	%.2f	The averages over the past 5, 15, and 60 minutes of the counts of messages currently enqueued to the channel
%counter_stored_messages_rate_5 %counter_stored_messages_rate_5_min %counter_stored_messages_rate_5_max %counter_stored_messages_acc_5 %counter_stored_messages_acc_5_min %counter_stored_messages_acc_5_max %counter_stored_messages_rate_15 %counter_stored_messages_rate_15_min %counter_stored_messages_rate_15_max %counter_stored_messages_acc_15 %counter_stored_messages_acc_15_min %counter_stored_messages_acc_15_max %counter_stored_messages_rate_60 %counter_stored_messages_rate_60_min %counter_stored_messages_rate_60_max %counter_stored_messages_acc_60 %counter_stored_messages_acc_60_min %counter_stored_messages_acc_60_max	float	%.2f	Rate of change (_rate_) and acceleration (_acc_), measured in messages per minute and messages per minute per minute, averaged over the past 5, 15, and 60 minutes. Also, the minimum and maximum values for these values.
%counter_stored_messages_threshold	int	%d	Stored message alarm threshold set for the channel in the option file.
%counter_stored_recipients %counter_stored_recipients_min %counter_stored_recipients_max	int	%d	Count of recipients currently enqueued to the channel and the minimum and maximum recorded counts.

Table 31–7 (Cont.) Substitution Strings for Use with the `show_counters` Command

Substitution string	Type	Format	Description
%counter_stored_volume_b %counter_stored_volume_min_b %counter_stored_volume_max_b %counter_stored_volume_k %counter_stored_volume_min_k %counter_stored_volume_max_k %counter_stored_volume_m %counter_stored_volume_min_m %counter_stored_volume_max_m %counter_stored_volume_g %counter_stored_volume_min_g %counter_stored_volume_max_g	float	%.2f	Count of volume currently enqueued to the channel as measured in bytes, kilobytes, megabytes, and gigabytes. Also, the minimum and maximum values recorded for these values.
%counter_submitted_messages	int	%d	Total cumulative count of messages sent by the channel (enqueued by the channel).
%counter_submitted_messages_rate_5 %counter_submitted_messages_rate_5_min %counter_submitted_messages_rate_5_max %counter_submitted_messages_acc_5 %counter_submitted_messages_acc_5_min %counter_submitted_messages_acc_5_max %counter_submitted_messages_rate_15 %counter_submitted_messages_rate_15_min %counter_submitted_messages_rate_15_max %counter_submitted_messages_acc_15 %counter_submitted_messages_acc_15_min %counter_submitted_messages_acc_15_max %counter_submitted_messages_rate_60 %counter_submitted_messages_rate_60_min %counter_submitted_messages_rate_60_max %counter_submitted_messages_acc_60 %counter_submitted_messages_acc_60_min %counter_submitted_messages_acc_60_max	float	%.2f	Rate of change (_rate_) and acceleration (_acc_), measured in messages per minute and messages per minute per minute, averaged over the past 5, 15, and 60 minutes. Also, the minimum and maximum values for these values.
%counter_submitted_recipients	int	%d	Total cumulative count of message recipients sent to by the channel (enqueued to by the channel).
%counter_submitted_volume_b %counter_submitted_volume_k %counter_submitted_volume_m %counter_submitted_volume_g	float	%.2f	Total cumulative count of message volume sent by the channel (enqueued by the channel) as measured in bytes, kilobytes, megabytes, and gigabytes.

31.7.3.3.2 .HELD messages: command=show_held

The `show_held` command may be used to determine whether or not there are `.HELD` message files. This command merely determines whether or not such files exist; it does not return a count of how many such files there are. On systems which have large volumes of queued messages, this command may take a while to execute. Parameters which may be supplied in conjunction with the command are listed in the following table.

Parameter		Description
held_format=*file-spec*	Required	Name of the formatting file to use to format .HELD message information. The recognized substitution strings for this command are listed in Table 31–6 and Table 31–8.
success_format=*file-spec*	Required	Name of the formatting file to use to format the results when the command is successful.
error_format=*file-spec*	Required	Name of the formatting file to use to format the results when the command fails.

Each of the *file-spec* file specifications must be relative file paths specifying files in the pmdf_root:[www.monitor] directory (OpenVMS) or the /pmdf/www/monitor/ directory (UNIX).

An example URL for a GET command might be

```
http://host:7633/monitor/?command=show_held&success_format=hsuccess.txt&
  error_format=herror.txt
```

Table 31–8 Substitutions Strings for Use with the show_held Command

Substitution string	Type	Format	Description
%held_messages	int	%d	Has value 0 when there are no .HELD message files and the value 1 when there are .HELD message files.

31.7.3.3.3 General Information: command=show_pmdf

The show_pmdf command displays general information about PMDF. Parameters which may be used with the command are listed below.

Parameter		Description
pmdf_format=*file-spec*	Required	Name of the formatting file to use to format PMDF information. The recognized substitution strings for this command are listed in Table 31–6.
success_format=*file-spec*	Required	Name of the formatting file to use to format the results when the command is successful.
error_format=*file-spec*	Required	Name of the formatting file to use to format the results when the command fails.

Each of the *file-spec* file specifications must be relative file paths specifying files in the pmdf_root:[www.monitor] directory (OpenVMS) or the /pmdf/www/monitor/ directory (UNIX).

An example URL for a GET command might be

```
http://host:7633/monitor/?command=show_pmdf&pmdf=pmdf.txt&
  success_format=psuccess.txt&error_format=perror.txt
```

31.7.3.3.4 Processing Queues: command=show_queue

The `show_queue` command displays information about OpenVMS processing queues and jobs. This command is only available on OpenVMS PMDF systems. The parameters recognized by the command are listed below.

Parameter		Description
`queue_format=`*`file-spec`*	Required	Name of the formatting file to use to format processing queue information. The recognized substitution strings for this command are listed in Table 31–6 and Table 31–9.
`job_format=`*`file-spec`*	Optional	Name of the formatting file to use to format processing job information. The recognized substitution strings for this command are listed in Table 31–6 and Table 31–10.
`queue_name=`*`name`*	Optional	Name of the processing queue to display information for. Wildcards are permitted. If this option is not specified, each processing queue used by PMDF will be scanned.
`success_format=`*`file-spec`*	Required	Name of the formatting file to use to format the results when the command is successful.
`error_format=`*`file-spec`*	Required	Name of the formatting file to use to format the results when the command fails.

Each of the *`file-spec`* file specifications must be relative file paths specifying files in the `pmdf_root:[www.monitor]` directory (OpenVMS) or the `/pmdf/www/monitor/` directory (UNIX).

For each queue matching the queue name pattern, the queue file and optional job format file will be used to format the information for that queue. It is important to note that the `%first`, `%last`, `%!first`, and `%!last` formatting strings are handled in a special way when both queue and job formatting files are specified. In that case, the first flag is only true for the first queue entry formatted by the queue file. It is not true while the job information is being formatted. The last flag is never true while the queue information is being formatted. It is only true when the last job of the last queue is being formatted.

An example URL for a GET command might be

```
http://host:7633/monitor/?command=show_queue&queue_format=queue.txt&
  job_format=job.txt&success_format=qsuccess.txt&error_format=qerror.txt&
  queue_name=MAIL$BATCH
```

Table 31–9 Queue Substitution Strings for Use with the `show_queue` Command

Substitution string	Type	Format	Description
`%queue_assigned_name`	string	%s	Name of the execution queue associated with the logical queue name.
`%queue_base_priority`	uint	%u	The priority at which at which batch jobs submitted to the queue are initiated or the priority of a symbiont process that controls output execution queues for the queue.

`%queue_cpu_default_s`
`%queue_cpu_default_m`
`%queue_cpu_default_h`
`%queue_cpu_default_d`
	uint	%.2f	The default CPU time limit, displayed in units of seconds, minutes, hours, or days, for jobs submitted to the queue.

`%queue_cpu_default_dhms`	uint	%u %02u:%02u:%02u	The default CPU time limit, displayed in "dd hh:mm:ss" format, for jobs submitted to the queue.

`%queue_cpu_limit_s`
`%queue_cpu_limit_m`
`%queue_cpu_limit_h`
`%queue_cpu_limit_s`
	uint	%.2f	The maximum CPU time limit, displayed in units of seconds, minutes, hours, or days, for jobs submitted to the queue.

`%queue_cpu_limit_dhms`	uint	%u %02u:%02u:%02u	The maximum CPU time limit, displayed in "DD HH:MM:SS" format, for jobs submitted to the queue.
`%queue_default_form_name`	string	%s	The name of the default form associated with the queue.
`%queue_default_form_stock`	string	%s	The name of the default paper stock associated with the queue.
`%queue_description`	string	%s	Description associated with the queue.
`%queue_device_name`	string	%s	The name of the device on which the specified output execution queue is located.
`%queue_executing_job_count`	uint	%u	Count of jobs currently being executed in the queue.
`%queue_flags`	uint	%s	Queue processing flags.
`%queue_form_name`	string	%s	Form name associated with the queue.
`%queue_form_stock`	string	%s	Paper stock associated with the queue.
`%queue_generic_target`	string	%s	The names of the execution queues which are enabled to accept from the queue.

Table 31–9 (Cont.) Queue Substitution Strings for Use with the `show_queue` Command

Substitution string	Type	Format	Description
%queue_holding_job_count	uint	%u	The count of jobs being held in the queue until explicitly released for execution.
%queue_job_limit	uint	%u	The number of jobs which can execute simultaneously in the queue.
%queue_job_reset_modules	string	%s	The names of the text modules that are to be extracted from the device control library and copied to the printer prior to each print job.
%queue_job_size_maximum	uint	%u	The maximum number of disk blocks that a print job initiated from the queue may contain.
%queue_job_size_minimum	uint	%u	The minimum number of disk blocks that a print job initiated from the queue may contain.
%queue_library_specification	string	%s	The name of the device control library for the queue.
%queue_name	string	%s	The name of the queue.
%queue_owner_uic	uint	%s	The UIC of the owner of the queue.
%queue_pending_job_block_count	uint	%u	The total number of blocks for all pending jobs in the queue. Applies only to execution queues.
%queue_pending_job_count	uint	%u	The total number of jobs pending execution in the queue.
%queue_processor	string	%s	The name of the symbiont image associated with the queue.
%queue_protection	uint	%s	The protection mask associated with the queue.
%queue_retained_job_count	uint	%u	The number of jobs in the queue retained after execution.
%queue_scsnode_name	string	%s	The name of the node on which the queue is located. Applies only to execution queues.
%queue_status	uint	%s	The status of the queue.
%queue_timed_release_job_count	uint	%u	The number of jobs in the queue on hold until a specified time.
%queue_type	uint	%s	A text string describing the type of queue.
%queue_wsdefault	uint	%u	The default working set size for jobs run in the queue. Applies only to batch and execution queues.

Table 31–9 (Cont.) Queue Substitution Strings for Use with the `show_queue` Command

Substitution string	Type	Format	Description
`%queue_wsextent`	uint	%u	The default working set extent for jobs run in the queue. Applies only to batch and execution queues.
`%queue_wsquota`	uint	%u	The default working set quota for jobs run in the queue. Applies only to batch and execution queues.

Table 31–10 Job Substitution Strings for Use with the `show_queue` Command

Substitution string	Type	Format	Description
`%job_account_name`	string	%s	Account name of the owner of the job.
`%job_after_time`	string	%s	Date and time when the job is scheduled to be released to run (/AFTER).
`%job_checkpoint_data`	string	%s	Job's checkpoint data stored in the BATCH$RESTART DCL symbol.
`%job_cli`	string	%s	The command language interpreter to be used to execute the job (/CLI).
`%job_completed_blocks`	uint	%u	Blocks completed so far by the symbiont for the job. Applies only to print jobs.
`%job_completion_queue`	string	%s	Name of the queue in which the job was executed.
`%job_completion_time`	string	%s	Date and time at which the job finished executing.
`%job_condition_vector`	uint	%u	Final completion status code for the job.
`%job_copies`	uint	%u	Number of copies to be printed by the print job (/COPIES). Applies only to print jobs.
`%job_copies_done`	uint	%u	Copies printed so far by the symbiont for the job. Applies only to print jobs.

`%job_cpu_limit_s`
`%job_cpu_limit_m`
`%job_cpu_limit_h`
`%job_cpu_limit_d`
	float	%.2f	CPU time limit allowed for the job displayed in seconds, minutes, hours, or days (/CPUTIME).

`%job_cpu_limit_dhms`	uint	%u %02u:%02u:%02u	CPU time limit allowed for the job displayed in "DD HH:MM:SS" format (/CPUTIME).
`%job_entry_number`	uint	%u	Queue entry number assigned to the job.
`%job_file_count`	uint	%u	Count of files submitted for the job.
`%job_flags`	uint	%s	Job processing flags.
`%job_form_name`	string	%s	Name of the print form associated with the job or queue (/FORM=). Applies only to print jobs.
`%job_form_stock`	string	%s	Name of the paper stock associated with to be used for the job. Applies only to print jobs.

Table 31–10 (Cont.) Job Substitution Strings for Use with the `show_queue` Command

Substitution string	Type	Format	Description
%job_log_queue	string	%s	Name of the print queue to which the log for the job is to be submitted for printing (/PRINTER).
%job_log_specification	string	%s	File specification for the job's log file (/LOG).
%job_name	string	%s	Name associated with the job (/NAME).
%job_note	string	%s	Note associated with the job (/NOTE).
%job_operator_request	string	%s	Request to be sent to the operator before the job begins execution (/OPERATOR).
%job_parameter_1 %job_parameter_2 %job_parameter_3 %job_parameter_4 %job_parameter_5 %job_parameter_6 %job_parameter_7 %job_parameter_8	string	%s	Parameters P1 through P8 associated with the job (/PARAMETER).
%job_pending_reason	uint	%s	Reason for why the job is pending execution.
%job_pid	uint	%08x	Process identifier of the process executing the job request.
%job_priority	uint	%u	Job processing priority (/PRIORITY).
%job_queue_name	string	%s	Name of the queue that contains the job (/QUEUE).
%job_requeue_queue_name	string	%s	Name of the queue to which the job has been reassigned.
%job_restart_queue_name	string	%s	Name of the queue to submit the job to should the job be restarted.
%job_retention_time	string	%s	The time until which the job should be retained in the queue.
%job_size	uint	%u	Size of the job as measured in 512 byte blocks (*e.g.,* size of the command procedure to be executed or the size of the file to be printed).
%job_status	uint	%s	Job status information.
%job_submission_time	string	%s	Date and time when the job was submitted to the queue.
%job_uic	uint	%s	UIC of the account under which the job was submitted.
%job_username	string	%s	Username of the account under which the job was submitted (/USER).
%job_wsdefault	uint	%u	Job's working set default (/WSDEFAULT).
%job_wsextent	uint	%u	Job's working set extent (/WSEXTENT).
%job_wsquota	uint	%u	Job's working set quota (/WSQUOTA).

31.7.3.3.5 HTML SUBMIT Buttons: noop=label

The noop=*label* pair is provided as an aid to processing HTTP POST requests generated with HTML forms. The need for this command relates to a shortcoming of HTML. In HTML, the form submit operation is realized through the following HTML tag:

```
<INPUT TYPE="SUBMIT" NAME="command" VALUE="y">
```

That tag generates the URL-encoded command command=*y* when the button labelled *y* is activated. But here's the rub: the label which appears on the button must be the same as the command to be submitted. So as to prevent HTML authors from having to label buttons to match the CGI's command names, the noop command is provided. It is used as follows

```
<INPUT TYPE="HIDDEN" NAME="command" VALUE="command-name">
<INPUT TYPE="SUBMIT" NAME="noop" VALUE="button-label">
```

The above allows whatever text label is desired to be placed on the submission button. When the submission button is generated a URL-encoded command of the form

```
noop=button-label&command=command-name&parameter-name-1=parameter-value-1...
```

is sent to the CGI. The CGI ignores the noop= *button-label* pair and processes the remainder of the URL encoded command.

31.7.3.4 An HTTP GET Example

The following example illustrates three formatting files—command-specific, success, and error—which might be used to display information about the PMDF channel counters. An HTTP request using these files might issue an HTTP GET for the URL

```
http://host:7633/monitor/?command=show_channels&counters_format=counters.txt
  &success_format=success.txt&error_format=error.txt
```

The command-specific formatting file is shown in Example 31–4. The file builds a table displaying each channel name (%counter_channel) and the count of messages currently enqueued to the channel (%counter_stored_messages). In addition, the cumulative count of messages enqueued to (%counter_received_messages), dequeued by (%counter_delivered_messages), and enqueued by (%counter_submitted_messages) the channel are also shown. The values for each PMDF channel are substituted into the HTML in place of the substitution strings which begin with %. The %first and %last substitution strings are used to generate HTML for the start and end of the table.

Once information for each channel is formatted, the success formatting file of Example 31–5 is then used to produce the final content of the HTTP response to be sent back to the client. Sample HTML output is shown in Example 31–7; that output builds a table appearing similar to the one shown below:

```
PMDF channel counters
            Current Messages          Cummulative Messages
                 Enqueued to   Enqueued to  Dequeued by  Enqueued by
 conversion                1         2300         2299           0
 l                        23        18021        17998       19473
 tcp_local               147        13522        13375         321
```

Should an error occur, the error formatting file of Example 31–6 will instead be used to format the response sent back to the client.

Example 31–4 `show_channels` **example:** `channels.txt` **formatting file**

```
%first{<TABLE BORDER><TR><TH><TH>Current Messages<TH COLSPAN=3>}
%first{Cummulative Messages<TR ALIGN="right"><TH><TH>Enqueued to}
%first{<TH>Enqueued to<TH>Dequeued by<TH>Enqueued by}
<TR ALIGN="right"><TD ALIGN="left">%counter_channel
<TD>%counter_stored_messages<TD>%counter_received_messages
<TD>%counter_delivered_messages<TD>%counter_submitted_messages
%last{</TABLE>}
```

Example 31–5 `show_channels` **example:** `success.txt` **formatting file**

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Refresh" CONTENT="60">
<TITLE>PMDF channel counters</TITLE>
</HEAD>
<BODY>
<B>PMDF channel counters</B>
<P>
%s
</BODY>
</HTML>
```

Example 31–6 `show_channels` **example:** `error.txt` **formatting file**

```
<META HTTP-EQUIV="Refresh" CONTENT="60">
<HTML>
<HEAD>PMDF Channel Counters</HEAD>
<BODY>

Unable to obtain information on the PMDF channels.  Output from the server is
shown below

<P>
<HR>
<P>
%s
<P>
<HR>
</BODY>
</HTML>
```

Example 31–7 `show_channels` **example: the resulting HTML output**

```
<HTML>
<HEAD>
<META HTTP-EQUIV="Refresh" CONTENT="60">
<TITLE>PMDF channel counters</TITLE>
</HEAD>
<BODY>
<B>PMDF channel counters</B>
<P>
<TABLE BORDER><TR><TH><TH>Current Messages<TH COLSPAN=3>
Cummulative Messages<TR ALIGN="right"><TH><TH>Enqueued to
<TH>Enqueued to<TH>Dequeued by<TH>Enqueued by
<TR ALIGN="right"><TD ALIGN="left">conversion
<TD>1<TD>2300
<TD>2299<TD>0
<TR ALIGN="right"><TD ALIGN="left">l
<TD>23<TD>18021
<TD>17998<TD>19473
<TR ALIGN="right"><TD ALIGN="left">tcp_local
<TD>147<TD>13522
<TD>13375<TD>321
</TABLE>
</BODY>
</HTML>
```

32 Performance Tuning

There are a variety of things which can be done to improve PMDF's performance. However, before trying to tune PMDF you should first feel comfortable with PMDF: have a basic understanding of how it works, be familiar with your configuration, and be able to recognize when PMDF isn't working on your system. In addition, it is important that you spend some time identifying what the bottlenecks are on your system: CPU resources, disk speed, memory, network speed or latencies, *etc.* Without a clear idea of where the bottlenecks are, any tuning you do is likely to be ineffective.

32.1 Basics

It is important that you understand some of the basics of how PMDF works. This section attempts to present a very basic overview.

When PMDF receives a message, PMDF writes the message as one or more disk files in the PMDF queue directories. These files represent copies of the message: at least one copy for each channel to which the message must be enqueued. It is crucial that the received message be written to a non-volatile medium such as a magnetic disk file: were the system to crash before the message could be sent on to its final destination, then the message might be lost. For this reason, PMDF always writes received messages to disk before giving a positive acknowledgement of receipt to the transmitter of the message. After a message is received, an entry is made in the queue cache database.

Once PMDF has received a message, it attempts to deliver it. This is done by submitting a processing job for each channel to which the message is enqueued. These jobs attempt to send the message to wherever it is next bound (as determined by PMDF's domain rewriting rules). If a job is successful, then the message copy it was handling is deleted and the corresponding entry removed from the queue cache database. If not successful, then the message copy is left on disk for a subsequent delivery attempt.

So, the normal mode of operation is: A message is received, it is written to a file, a record is added to the queue cache database, a processing job is started, the job reads the message file, the job then deletes the file, the record is removed from the queue cache database.

Given this basic scenario, several things should be clear:

- Increased throughput can be realized by:
  - + decreasing disk write and read times,
  - + increasing the internal memory buffer size for processing jobs to decrease the use of temporary buffer files for large messages,
  - + increasing the number of simultaneous processing jobs and any resources they might require,

  + decreasing processing job overhead,
  + decreasing per message processing job overhead by increasing the number of messages handled per job,
  + ensuring that the number of SMTP server processes available for accepting incoming SMTP over TCP/IP messages is appropriate for the level of message traffic,
  + for general SMTP over TCP/IP channels, used to send SMTP messages to multiple different destinations, decreasing per connection overhead for outgoing SMTP over TCP/IP messages by collecting and sending messages to the same destination host in one connection,
  + for `daemon` SMTP over TCP/IP channels, commonly used to send SMTP messages to single specific relay systems such as mailhubs or firewalls, using multiple threads for outgoing connections, and
  + tuning the queue cache database.
- Using a virtual RAM disk for the message store is a very bad idea. Should your system crash, mail can be lost or corrupted.[1]
- Keeping the message store on a shadowed disk can hurt performance: whereas shadowset reads are on the average faster, writes are on the average slower. Since usually only one read will be required, the decreased read time will not be sufficient to compensate for the increased write time.

The suggestions in the first bullet item above, as well as several others, are explored in the remainder of this chapter.

VMS

On OpenVMS, note that the queue cache database is an RMS keyed, indexed file. As such, it can be tuned using any of the standard RMS tuning tools. Should you want to tune it differently from the FDL parameters in the file `PMDF_COM:queue_cache.fdl`, you might first consult with Process Software.

32.2 CPU and Resources

First off, if you have not begun to use a compiled configuration, then begin doing so. This will noticeably reduce the startup time of PMDF processing jobs (and on OpenVMS, the startup time for PMDF MAIL) as well as reduce the time spent waiting for a response the first time your users use a PMDF handled address in their user agent, such as send a message from Pine (or on OpenVMS, send to an IN% address in VMS MAIL). See Section 8.1 for instructions on how to generate a compiled configuration.

Consider establishing processing queues (OpenVMS) or Job Controller queues (UNIX and NT) for specific channels which you want to ensure always have processing slots. For instance, set up a separate queue for your pager channel so that delivery jobs for urgent pages do not get held up waiting in the MAIL$BATCH queue (OpenVMS)

[1] With certain provisos, however, storing the queue cache database on a virtual RAM disk can be safe enough — see the discussion later in this chapter. And a reliable, battery-backed, solid-state RAM disk can be safe enough for the message store.

or DEFAULT queue (UNIX and NT). Then use the queue keyword to direct particular channels to run in particular queues; see Section 2.3.4.18 for more details on the queue keyword, and Section 32.4 for a further discussion of directing channels to run in specific queues.

Busy channels — channels which usually have immediate jobs processing in the queues — are likely to achieve greater overall throughput in exchange for slightly increased latency by use of the after channel keyword with a (typically small) delta time value. By specifying a delta time value that, while not introducing too much of a delay for new messages, does allow PMDF to "collect" multiple messages to be handled by one channel job, the overhead of image activation or expensive protocol connections can be reduced. For a busy channel, this can lead to a substantial increase in overall throughput. Channels such as multithreaded TCP/IP channels, or on OpenVMS also the L channel or MR channels, are often candidates. Multithreaded TCP/IP channels, for instance, sort messages to different hosts into different threads; when given multiple messages to deliver to a single host, those messages can then be delivered during a single SMTP connection session. See Section 2.3.4.18 for more details on the after keyword.

If your system has the memory to spare, increasing the size of message that processing jobs can buffer internally can reduce use of temporary buffer files on disk when receiving or processing large messages. See the discussion of the MAX_INTERNAL_BLOCKS PMDF option in Section 7.3.5. On OpenVMS, make sure that the account under which PMDF jobs are operating (normally the SYSTEM account) has sufficient memory quotas; PGFLQUOTA, WSDEF, WSQUO, and WSEXTENT are particularly relevant.

The PMDF Dispatcher controls the creation and use of multithreaded SMTP server processes. If, as is typical, incoming SMTP over TCP/IP messages are a major component of e-mail traffic at your site, monitor how many simultaneous incoming SMTP connections you tend to have, and the pacing at which such connections come in. Tuning of Dispatcher configuration options controlling the number of SMTP server processes, the number of connections each can handle, the threshold at which new server processes are created, *etc.*, can be beneficial if your site's incoming SMTP over TCP/IP traffic is unusually high or low.

For typical SMTP over TCP/IP channels, used to send to multiple different remote systems, the PMDF multithreaded TCP/IP channel's default behavior of sorting messages to different destinations into different threads and then handling all messages to a single host in a single thread is desirable for performance. However, for a daemon TCP/IP channel, one dedicated to sending to a specific system, if the receiving system supports multiple simultaneous connections it can be preferable to force PMDF to split the outgoing messages into separate threads, by using a combination of the threaddepth keyword set for some appropriate value and the MAX_CLIENT_THREADS channel option; see Section 2.3.4.29 and Section 21.1.2.2.

VMS

To make better utilization of your CPU resources, consider making MAIL$BATCH a generic queue feeding specific queues across your cluster. If you do this, keep in mind that those channels which use software available on only a few systems must do their processing on those systems. For instance, if you only run Jnet on one system, then you must process your bit_ channels on that system. Use the queue channel keyword to designate which queues a channel should use for its processing. By default, channels

will use MAIL$BATCH; thus, you need only specify this keyword on those channels which should use a separate queue. See 2.3.4.18 for information on the `queue` keyword.

On OpenVMS, if you are not already using the Process Symbiont, then consider using it. By default, MAIL$BATCH is a batch queue. Thus, PMDF's processing jobs are by default batch jobs: each processing job must be created and go through the LOGINOUT procedure. The Process Symbiont reduces this overhead by instead using a pool of detached processes for PMDF processing. When PMDF needs to launch a processing job, one of the idle detached processes is used. This avoids the overhead of creating and logging in a new process for each processing task. Use of Process Symbiont queues also reduces the creation of unnecessary log files. The Process Symbiont is a multi-threaded server symbiont. You can control how many detached processes it runs and how long they are allowed to remain idle before being deleted. As with batch jobs, you can create a generic Process Symbiont queue which feeds specific queues spread across your cluster. See Section 9.1 for instructions on how to configure the Process Symbiont.

On OpenVMS, installing as known images channel programs for often used channels will reduce processing job overhead. For local delivery, you do not need to do anything: the requisite images, `SYS$SYSTEM:mail.exe` and PMDF_SHARE_LIBRARY, are already installed. Sites with large volumes of outgoing SMTP messages should consider installing the `PMDF_EXE:tcp_smtp_client.exe` image (using the DCL INSTALL utility's /OPEN/HEADER/SHARED qualifiers).

32.3 Disks and Files

The most common bottleneck in PMDF is disk I/O. PMDF does a lot of it. Try to keep the disks with PMDF's message store below 66% capacity so that the operating system can efficiently manage file create and delete cycles. This is especially important on OpenVMS where the file system begins to become very inefficient once the disk gets over 66% capacity. Also, use disk striping or other aggregate disk spindle techniques that help both read and writes. Avoid disk shadowing if possible. *Disk is cheap these days: spend money on multiple spindles and sufficient free space.*

By using symbolic links under the `/pmdf` directory (UNIX), by redefining the PMDF_QUEUE and PMDF_LOG logicals (OpenVMS), or by redefining the PMDF_QUEUE and PMDF_LOG registry entries (Windows) you can redirect where PMDF keeps its message store and log files. PMDF's command, executable, and table directories[2] can also be separated if absolutely necessary.

The location for PMDF temporary files can also be moved. PMDF_SCRATCH controls the location of temporary unnamed files (such as those used to buffer incoming large SMTP messages or incoming large messages submitted by local users); PMDF_TMP controls the location of temporary named files (such as those used by the conversion channel). On UNIX, both values default to `/pmdf/tmp` if not explicitly pointed elsewhere in the PMDF tailor file. Similarly on Windows, both values default to `C:\pmdf\tmp`

[2] On OpenVMS, these are `PMDF_COM:`, `PMDF_EXE:`, and `PMDF_TABLE:`, respectively; on UNIX, these are `/pmdf/lib`, `/pmdf/bin`, and `/pmdf/table`, respectively; on Windows, these are pointed to by the registry entries `PMDF_COM`, `PMDF_EXE`, and `PMDF_TABLE` respectively.

if not explicitly pointed elsewhere by in the Windows registry. On OpenVMS, if PMDF_SCRATCH and PMDF_TMP logicals are not defined then temporary unnamed files default to SYS$SCRATCH (next SYS$DISK, next SYS$LOGIN) and temporary named files default to `PMDF_QUEUE:[000000]`. Note that if explicitly defining PMDF_SCRATCH it is important to point it to a device on which any user can create files.

On OpenVMS only, the location for files usually put into PMDF_TMP can be broken up. If the logical name PMDF_IMAP_TMP is defined, then the legacy IMAP server will put its temporary files in the directory named by that logical, rather than in PMDF_TMP. If the logical name PMDF_SPEC_TMP is defined, then channels who create special temporary files (such as the `conversion`, `script`, and `pipe` channels) will put their temporary files in the directory named by that logical rather than in PMDF_TMP.

By default, the messages for a given channel are stored in a single, channel-specific directory under `PMDF_QUEUE:` (OpenVMS) or `/pmdf/queue/` (UNIX) or usually `C:\pmdf\queue\` (Windows). File system performance degrades rapidly for directories with more than a couple thousand files; this can present a problem for channels which see heavy message traffic — especially when the network associated with that channel is down and messages begin to queue up. Use the `subdirs` channel keyword to indicate that a channel should uniformly spread its messages across several subdirectories. For Internet sites with heavy traffic loads, this should be done for their outgoing TCP/IP channel, usually tcp_local.

By changing the PMDF_QUEUE_CACHE_DATABASE logical (OpenVMS only), you can move the queue cache database to an alternate location. After moving it, be sure to issue the OpenVMS commands

```
$ PMDF CACHE/CLOSE
$ PMDF CACHE/SYNCH
```

so as to ensure that all PMDF processing jobs close the old database and begin using the new, relocated database.

It is safe to keep the queue cache database on a virtual RAM disk provided that:

1. You periodically copy the queue cache database to a regular disk. (On OpenVMS, use BACKUP/IGNORE=INTERLOCK to make a copy of the queue cache database on a running system.)
2. You resynchronize the queue cache database each time you reboot your system. On OpenVMS, use the command PMDF CACHE/SYNCHRONIZE. The cache resynchronization should be performed after executing the `pmdf_startup.com` procedure (on OpenVMS) and before starting your networks or PMDF processing queues.

High volume sites that have enabled the `logging` channel keyword to log message traffic, as well as the LOG_CONNECTION PMDF option to log connections, can want to enable the SEPARATE_CONNECTION_LOG option to direct the connection log entries to a separate file from the message traffic log entries; see Section 7.3.6. On OpenVMS, busy sites that have enabled the `logging` keyword can also find it beneficial to increase the LOG_ALQ and LOG_DEQ options, also discussed in Section 7.3.6, to use larger file extents for the underlying file allocation.

UNIX and Windows sites can consider whether for their use it is acceptable to set the PMDF option FSYNC=0. Doing so improves performance, but at the cost that if a UNIX or Windows system crashes at just the wrong moment, messages not yet synched to disk could be lost. See Section 7.3.8.

32.4 Planning for Network Speed and Latency

While you can not be able to speed up your network links or reduce network latencies, you can at least arrange your message processing so as to recognize these issues. For instance, Internet sites can want to have two separate outbound TCP/IP channels: one for intra-domain traffic and the other for inter-domain traffic. Suppose, for instance, that your domain is claremont.edu. Then you might set up a new PMDF_SMTP_CLAREMENT generic or execution queue (OpenVMS) or Job Controller queue (UNIX and NT). Next add the rewrite rule

```
.claremont.edu          $U%$H$D@TCP-CLAREMONT
```

and the channel definition

```
tcp_claremont smtp single_sys nox_env_to mx queue PMDF_SMTP_CLAREMONT
TCP-CLAREMONT
```

to your PMDF configuration. This will cause all mail to other hosts in the claremont.edu domain to go out through the tcp_claremont channel. Processing jobs will run in the PMDF_SMTP_CLAREMONT processing queue. Internet mail for other domains will continue to go out the tcp_local channel and use the MAIL$BATCH queue (OpenVMS) or DEFAULT Job Controller queue (UNIX and NT).

You can apply this concept to other networks and channels. Indeed, you might find it useful to use separate processing queues for each of your outbound channels. This way, when one network is particularly slow, its processing jobs do not hold up jobs from other channels which can be waiting for the next available processing slot in the processing queue.

32.5 Differential Handling of Large or Low Priority Messages

For some sites, processing all messages, of whatever sort, as quickly as possible is the major goal, and issues of message size or message priority are not a concern.

Other sites can want to treat messages over some site-specified size as deserving of lower priority or delayed processing, as compared with messages of a more “normal” size.

PMDF by default pays attention to message priority when processing more than one message (as for a PMDF periodic delivery job) and will automatically try to send messages of greater priority before trying to send messages of lower priority.

On OpenVMS, to get generally different handling of messages of differing priority, the `nonurgentqueue`, `normalqueue`, and `urgentqueue` channel keywords can be used to cause messages of differing priority to be processed in different processing queues; see Section 2.3.4.18. Such different processing queues might have different resources devoted to them, for instance, different job limits as to how many simultaneous jobs can run at once; or a processing queue used for processing non-urgent messages might be set up to run only at off-hours.

PMDF can be configured to treat messages over site-specified sizes as having an effectively lower processing priority via the `nonurgentblocklimit`, `normalblock-limit`, and `urgentblocklimit` channel keywords, or the NON_URGENT_BLOCK_LIMIT, NORMAL_BLOCK_LIMIT, and URGENT_BLOCK_LIMIT PMDF options; see Section 2.3.4.10 or Section 7.3.5, respectively. In conjunction with differential priority handling, as described above, this can provide for differential handling of messages of varying sizes.

33 Maintenance and Troubleshooting on OpenVMS

Under normal conditions PMDF accepts, processes, and promptly delivers messages. If a message can not be delivered immediately, PMDF will periodically retry the delivery. Intervention by a system or network manager is generally not required, as PMDF is designed to perform its functions unattended as much as possible. It is a good idea, however, to periodically inspect the PMDF queue directories. System or network problems over which PMDF has no control can adversely affect normal operation. In addition, unusual behavior from PMDF can often point out network problems which can otherwise go unnoticed.

See also Chapter 31, which discusses monitoring PMDF activity. A baseline for what normal PMDF activity looks like at your site can be helpful when questions arise.

33.1 Background on PMDF Operation

In order to diagnose and repair PMDF message delivery problems, you need some understanding of the steps that PMDF goes through to process and deliver a message. Each time a message is handed to PMDF, a file is created in the appropriate message queue subdirectory. Each channel has a corresponding subdirectory. The subdirectory into which the message is placed is that of the destination channel which will handle the message next. For example, a message coming in from any source to be delivered to a local user will be placed in the subdirectory corresponding to the local, or L, channel. A message coming in sent by a local user's POP or IMAP client to be routed on through SMTP over TCP will be placed in the subdirectory for the outbound TCP/IP channel, such as tcp_local.

Once the message is stored in the appropriate queue subdirectory, a PMDF processing job is usually submitted to the appropriate batch or server queue. This step is omitted for channels marked with the `periodic` keyword. The processing job will attempt to deliver the message. If it cannot be delivered and the failure is judged to be a permanent condition, the message is returned to the sender and the local postmaster. If the failure is a temporary error condition, the message is left in the message queue to be retried later.

Note that messages are processed by jobs submitted through the OpenVMS batch and print queue subsystem. PMDF jobs can be executed as batch jobs through standard OpenVMS batch queues or as detached processes through the PMDF Process Symbiont described in Chapter 9. Either way, you can use standard OpenVMS utilities such as SHOW QUEUE to monitor message processing.

For additional discussion of PMDF operation, see also Chapter 1.

33.2 Standard Maintenance Procedures

In tracking down problems with PMDF you should first determine whether the problem occurs before or after a message is entered into the message queue. If messages do not get placed in a queue directory at all, then such messages were never accepted: configuration problems, or environmental problems (*e.g.*, disk space or quota problems), or the absence of PMDF servers (such as the Dispatcher and its services) can prevent PMDF from accepting messages; or network connectivity or routing problems could mean that the messages are stuck or misdirected on a remote system. If messages are accepted but get put into the wrong queue directory, you probably have a configuration problem.

To track an errant mail message, you can start by examining each step of the process for errors. The subsections below discuss investigating these processing steps.

33.2.1 Check the PMDF Configuration

Use the PMDF TEST/REWRITE utility, described in Chapter 29, to test the response of your configuration to addresses. Certain basic sorts of problems in the PMDF configuration, such as clear syntax errors in the PMDF configuration, will cause the utility to issue an error message. Otherwise, the utility will show address rewrites that will be applied as well as the channel to which messages would ultimately be queued. If the output is not what you expect, you can need to modify your configuration.

33.2.2 Check Message Queue Directories

Check whether messages are present under the PMDF message queue directory, `PMDF_QUEUE:`. DCL commands such as DIRECTORY or the PMDF QM/MAINT utility's DIRECTORY command can be used to check for the presence of expected message files under the PMDF message queue directory.

If the PMDF TEST/REWRITE output looks correct but messages are not placed in the PMDF message queue directory, you can have a problem with disk quotas on the device holding your PMDF directories. Verify that disk quotas are disabled or sufficient quota has been allocated to the [SYSTEM] or [PMDF] UIC's. Check that the disk has not exhausted free space or file headers. Problems can also be due to an improper installation or startup of PMDF. Verify that `SYS$STARTUP:pmdf_startup.com` has been executed such that PMDF logical names are defined in the SYSTEM table in EXECUTIVE mode, and appropriate PMDF images have been installed as known files.

33.2.3 Check the Ownership of Critical Files

If you have a PMDF server account, usually called PMDF[1] and created during installation of PMDF, the PMDF queue and log directories and all subdirectories thereof and files contained therein must be owned by the PMDF server account. The PMDF queue and log directories are referenced via the logical names PMDF_QUEUE and PMDF_LOG, respectively, and are usually `PMDF_ROOT:[000000]queue.dir` and `PMDF_ROOT:[000000]log.dir`, respectively.

The queue cache database, `PMDF_TABLE:queue_cache.dat`, should also be owned by that same account. If the protection and ownership are not correct for the queue cache database, messages can not be entered into the queue cache, and the queue cache will become desynchronized. The proper protection of these files should be:

```
Directory PMDF_ROOT:[000000]

QUEUE.DIR;1           [PMDF]                  (RWED,RWED,,)
LOG.DIR;1             [PMDF]                  (RWED,RWED,,)

Directory PMDF_ROOT:[TABLE]

QUEUE_CACHE.DAT;1     [PMDF]                  (RWED,RWED,,)
```

If you do not have the PMDF server account, these files should have the same protection as above, but they should be owned by SYSTEM.

33.2.4 Check that the PMDF Dispatcher and Servers Are Present

Some PMDF channels, such as PMDF's multithreaded TCP/IP channels or Lotus Notes channels, include resident server processes that process incoming messages—such servers handle the slave (incoming) direction for the channel. In addition, remote clients using protocols such as POP and IMAP download their messages by connecting to PMDF's POP or IMAP servers.

The PMDF Dispatcher handles the creation of such PMDF servers. Dispatcher configuration options control whether such servers are available at all, and if available, how many such servers are created and when, and how many connections each server can handle. Most sites, for instance, will choose to have the Dispatcher always keep at least one SMTP server process resident.

The `PMDF PROCESS` utility can be used to show current PMDF processes, including PMDF server processes as well as currently executing PMDF channel jobs. Or the DCL command `SHOW SYSTEM/NET` can be used to show network processes, which will include such PMDF servers, as in the sample output shown in Figure 33–1. Except for PMDF sites that do not use any Dispatcher services, for all other PMDF sites the Dispatcher, indicated in the figure as ❶, should always be present during normal operation of PMDF.

[1] The PMDF server account is referenced by the `PMDF_SERVER_USERNAME` logical name.

Figure 33–1 Output of SHOW SYSTEM/NET

```
OpenVMS V6.2-1H2  on node THOR  15-NOV-2012 15:24:03.22  Uptime  5 08:31:23
  Pid    Process Name    State  Pri      I/O       CPU        Page flts  Pages
22A01A23 <DISPATCHER-01> HIB      6   201411   0 00:03:33.05    103992    389 N ❶
22A0271A <POP3-01>       HIB      6     9207   0 00:00:08.55     21729    389 N ❷
22A01A69 <LNMASTER-02>   HIB      6     9736   0 00:00:05.81      9918    296 N ❸
22A01FE2 <LNSLAVE-01>    HIB      6     5917   0 00:00:08.23      1545    606 N
22A02691 <IMAP-01>       HIB      6    57475   0 00:01:45.56      2940   1033 N ❹
22A025D1 <IMAP-02>       HIB      6    67728   0 00:01:30.59      1859   1278 N
22A029A1 <SNADS-02>      HIB      6      146   0 00:00:00.79      5849     76 N ❺
22A029A2 <HTTP-02>       HIB      6    12367   0 00:00:03.36      6135     83 N ❻
22A024A3 <WEB500-02>     HIB      6      156   0 00:00:00.75      6181     82 N ❼
22A028A5 <SMTP-01>       HIB      6    35128   0 00:01:03.10     22624    501 N ❽
```

❶ The Dispatcher should, at most PMDF sites, always be present. It controls the creation of PMDF server processes.

❷ One or more POP3 server processes can be present at sites using the PMDF POP3 server.

❸ Sites using PMDF-LAN Lotus Notes channels should have one LN master and one LN slave process present per Lotus Notes channel.

❹ One or more IMAP server processes can be present at sites using the PMDF IMAP server.

❺ Sites that are using the PMDF HTTP server to serve out PMDF documentation or PMDF monitoring information can have an HTTP server process present.

❻ Sites that are using the PMDF Web500 server to provide web browser access to an X.500 directory can have a Web500 server process present.

❼ Sites using the PMDF multithreaded TCP SMTP channel can have an SMTP server process present. Indeed, most such sites will want the Dispatcher configured so that there is always at least one SMTP server process present.

If the PMDF Dispatcher has been configured but is not present, it can be started with the command

```
$ PMDF STARTUP DISPATCHER
```

If the PMDF Dispatcher is present but some expected server is not present, check whether the Dispatcher has been configured to handle that service and if so, what the MIN_PROCS option value is for that service. For services with a MIN_PROCS value of 0, the server will only be created and present at times when a local client or remote site actually has a connection open; but for services with a MIN_PROCS value greater than or equal to 1, there should always be at least one such server process present.

See also Section 11.7 for a discussion of monitoring what connections are currently active to the Dispatcher.

33.2.5 Check that processing Jobs Start

If messages are enqueued correctly (received into the PMDF_QUEUE: area) but not delivered, you should first check that processing jobs are being submitted and executed on the correct batch or Process Symbiont queue.

Channel jobs by default are submitted to the MAIL$BATCH queue; most sites choose to configure the MAIL$BATCH queue as a generic queue feeding several execution queues. (Some sites can also choose to use alternate queues in addition to MAIL$BATCH. So normally, you should expect to see channel jobs executing in MAIL$BATCH, or its execution queues if MAIL$BATCH is a generic queue, and perhaps additional channel jobs pending in MAIL$BATCH if all queue processing slots are already occupied processing other channel jobs. And if you are using additional queues, you should expect to see channel jobs moving through those other queues as well.

The DCL SHOW QUEUE command can be used to show any currently processing jobs in execution queues, or the PMDF PROCESS utility will also show currently executing PMDF jobs (as well as resident PMDF server processes, as discussed in Section 33.2.4 above).

When a message is moving through PMDF, you can stop a processing queue with a command such as

```
$ STOP/QUEUE/NEXT MAIL$BATCH
```

and then look to see if the correct job has been submitted. If a job has not been submitted or the job exits immediately without attempting delivery, then verify that the queue name matches that listed by PMDF TEST/REWRITE. If accounting is enabled on your system, then you can use the ACCOUNTING utility to display the exit status of the batch job. Problems can be due to improper definition of PMDF logical names or incorrect ownership of the PMDF_LOG: directory. The directory should be owned by the PMDF account, if you have one, or the SYSTEM account if you do not.

33.2.6 Check Processing Log and Debug Log Files

If batch or Process Symbiont jobs are submitted and execute properly but the message stays in the message queue directory, you can examine the log files to see what is happening. All log files are created in the directory PMDF_LOG:. Log file name formats for various PMDF processing jobs are shown in Table 33–1.

Table 33–1 PMDF Log Files on OpenVMS

File name	Log file contains
*channel-name*_master.log	Output of master program (usually client) for *channel-name*
*channel-name*_slave.log	Output of slave program (usually server) for *channel-name*
dispatcher.log	Dispatcher logging if the Dispatcher DEBUG option has been set
*server-name*_server.log	Server process errors for *server-name*
*server-name*_thread.log	Per thread errors for *server-name*
task_server_*procsmb-queue*.log	Errors encountered by the *procsmb-queue* Process Symbiont queue
post.log	Log output for the periodic PMDF delivery job
return.log	Log output for the periodic PMDF message bouncer job

Channel log files are not created unless an error occurs or unless debugging output is enabled for the channel with the master_debug channel keyword or slave_debug channel keyword. See Section 2.3.4.85 for more information. Server log files are created when a new server process is created; if no error occurs they will contain merely normal OpenVMS login/out type information. Process Symbiont log files are created when a Process Symbiont queue is started; if no error occurs they will contain merely normal OpenVMS login/out information.

Note that version limits **must not be set** on files in the PMDF_ROOT: tree and particularly must not be set on files in the PMDF_LOG: directory. PMDF will take care of purging back old versions of log files itself, as one of its periodic housekeeping tasks, so there is no need for version limits—and version limits will cause a variety of problems.

33.2.7 Running a Channel Program Manually

While diagnosing a PMDF delivery problem it can be useful to run a PMDF delivery job by hand, particularly after enabling debugging output for one or more channels; see Section 2.3.4.85 for details on enabling debugging. The command

```
$ @PMDF_COM:submit_master channel-name
```

will submit a batch or Process Symbiont job to attempt outbound delivery for the channel *channel-name*. This is the same submit logic used when messages are accepted from VMS MAIL by PMDF for immediate delivery. If debugging is enabled for the channel in question, then submit_master.com will create a log file in PMDF_LOG: as described in Table 33–1.

The command

```
$ @PMDF_COM:master channel-name
```

will perform outbound delivery for the channel *channel-name* in the currently active process, with output directed to your terminal. This can be more convenient than submitting a job, particularly if you suspect problems with job submission itself.

In addition to *channel-name*, both `master.com` and `submit_master.com` will accept several other parameters:

```
$ @PMDF_COM:master channel-name [poll [since-time]]
$ @PMDF_COM:submit_master channel-name [poll [queue-name [since-time]]]
```

where the parameters are as listed in Table 33–2.

Table 33–2 `master.com` **and** `submit_master.com` **parameters**

channel-name	Required parameter specifying the channel. This determines which messages will be attempted and what channel master program will be executed. Channels known to your PMDF configuration appear as channel block names in your `pmdf.cnf` file.
poll	Optional parameter is a keyword specifying either `poll`, `nopoll`, or `ignore`. `nopoll` is the default and avoids executing the channel program if there are no messages awaiting outbound delivery. If the channel supports it, `poll` will execute the channel master program to pick up messages from a remote slave. `ignore` is used only by the G3_TO_FAX channel and causes time stamp checking to be bypassed.
queue-name	Optional parameter specifying the queue for submission. Applies to `submit_master.com` only.
since-time	Optional parameter specifying an OpenVMS date/time. Only messages queued after this date/time will be processed.

33.2.8 Checking that Periodic Jobs Are Present

In addition to immediate channel jobs created automatically when messages are first submitted into PMDF, or to manually created delivery jobs as described above in Section 33.2.7, PMDF also has periodic jobs that perform clean up tasks, and retry delivery of previously undelivered messages.

The command

```
$ SHOW QUEUE/ALL MAIL$BATCH
```

lists all jobs pending and executing in the default PMDF processing queue, MAIL$BATCH. Under idle conditions the command should result in output similar to that seen in Figure 33–2 or Figure 33–3.

Figure 33–2 Output of SHOW QUEUE/ALL MAIL$BATCH on a Basic PMDF System

```
$ SHOW QUEUE/ALL MAIL$BATCH

Generic server queue MAIL$BATCH

  Entry  Jobname         Username  Blocks  Status
  -----  -------         --------  ------  ------
    153  PMDF delivery   SYSTEM        11  Holding until 18-NOV-2012 15:13 ❶
     56  PMDF message bouncer
                         SYSTEM        12  Holding until 19-NOV-2012 00:30 ❷
```

Figure 33–3 Output of SHOW QUEUE/ALL MAIL$BATCH on a System With Optional Jobs

```
$ show queu/all mail$batch
Generic server queue MAIL$BATCH

  Entry  Jobname         Username     Blocks  Status
  -----  -------         --------     ------  ------
     95  PMDF delivery   SYSTEM           14  Holding until 18-NOV-2012 15:52:00 ❶
    810  PMDF message bouncer
                         SYSTEM           14  Holding until 19-NOV-2012 00:30:00 ❷
    399  PMDF PC Post    SYSTEM            8  Holding until 18-NOV-2012 13:05:03 ❸
    811  PMDF popstore message bouncer
                         SYSTEM            5  Holding until 19-NOV-2012 00:30:00 ❹
```

The two holding jobs indicated in the figures as ❶ and ❷ should always be present during normal operation of PMDF. They run periodically and resubmit themselves automatically. If you are using the PMDF popstore, you should also see a popstore return job, ❹, either holding or running (perhaps in an execution queue, if MAIL$BATCH is a generic queue). If you are using PMDF-LAN channels, then depending on just how you have configured it you can also see an additional job, the PMDF PC post job, ❸; if you are using DECUS UUCP and have defined the `PMDF_DO_RETURN_VN` logical so that `pmdf_submit_jobs.com` also submits the DECUS UUCP message bouncer job, then you should see that job as well.

❶ The "PMDF delivery" job is responsible for scanning the message queues and retrying delivery of messages waiting in the queues. The period defaults to four hours and can be changed with the `PMDF_POST_INTERVAL` logical name.

If the PMDF delivery job is lost, messages that are not immediately deliverable will tend to collect in the queue and never be retried. In addition, channels marked with the `periodic` channel keyword, which defers immediate delivery, will not function reliably.

❷ The "PMDF message bouncer" job checks the message queues for messages which do not appear to get through in a timely fashion and generates warning messages to the sender as well as the Postmaster. If such a message sits long enough, the PMDF message bouncer will eventually return the message to the sender and remove it from the message queue. This job usually runs once a day at 12:30 AM. If PMDF is configured for an hourly return interval (see Section 1.4.4) then the PMDF message bouncer will run every hour. The number of intervals between warnings is controlled by the `notices` channel keyword.

If the PMDF message bouncer job is not running, messages that cannot be delivered will sit in the message queue indefinitely and no notification will be sent to either the sender or the local Postmaster.

❸ The "PMDF PC Post" job is responsible for periodically polling for incoming messages from PC-LAN mail systems such as cc:Mail, Microsoft Mail, GroupWise (WordPerfect Office), and Novell MHS mail systems through PMDF-LAN channels, as well as for re-trying delivery of messages to those mail systems. Some PMDF-LAN configurations will use such a job; some will manage the running of PMDF-LAN channel jobs using other mechanisms.

❹ The "PMDF popstore message bouncer" job deletes old, stored PMDF popstore messages which have exceeded their alloted storage time, and sends notification messages back to the senders of messages which were never downloaded by their recipients. The job usually runs once a day around midnight. See the *PMDF popstore & MessageStore Manager's Guide* for full details.

If one or both of the PMDF delivery job and the PMDF message bouncer job is not present in the queue, (or if one of the other periodic jobs that your site uses is missing), it should be resubmitted using the `pmdf_submit_jobs.com` procedure

```
$ @SYS$STARTUP:pmdf_submit_jobs.com
```

The procedure is safe to run at any time, as it will not submit a job that is already present. As indicated in the OpenVMS Edition of the *PMDF Installation Guide*, Process Software recommends that `pmdf_submit_jobs.com` be executed as part of your normal system startup procedure to ensure that the basic PMDF periodic jobs are properly submitted and holding.

33.3 General Error Messages

There are a number of general sorts of issues that can interfere with the running of a variety of PMDF components; such issues include general syntax errors in a PMDF configuration, or license problems, or disk/quota problems leading to trouble writing files. The following sections describe some of the more common general error messages.

Note that the PMDF TEST/REWRITE utility will give warnings of many such common issues and with more detailed error messages than some other components of PMDF can display. So PMDF TEST/REWRITE can be a useful diagnostic tool: see if it is getting an error and if so, what.

Specific PMDF components can also issue other error messages, specific to that component. So when you an encounter an error not described below, see also the documentation on the PMDF component in question.

33.3.1 Errors in mm_init, such as "No room in ..." Errors

An "Error in mm_init" error generally indicates a PMDF configuration problem. Thus the PMDF TEST/REWRITE utility, which is often used to test the health of a PMDF configuration, can return such an error, as can other utilities such as PMDF CNBUILD. PMDF channels or servers can also return such errors, as can user agents attempting to submit messages to PMDF.

In particular, one of the more commonly encountered sorts of mm_init errors is a "No room in table" error or similar "No room in ..." sort of error. Generally, "no room in" errors are an indication that your current PMDF configuration has not set internal table sizes sufficient for the size of your PMDF configuration, and that it is time to have PMDF resize its internal tables, as described in Section 8.1.4.[2] However, some particular such "no room in ..." error messages can have alternate causes, and such cases are called out below. Any other "no room in" errors not explicitly mentioned are most likely simply an indication of a need to resize internal tables.

Rather than manually calculating and setting table sizes, you should use the PMDF CNBUILD utility to automatically resize the tables for you. See Section 8.1.4 for exact instructions on doing this.

If you use a compiled configuration, then be sure to recompile your configuration and reinstall it after resizing the tables.

If you're using the PMDF multithreaded SMTP server, the PMDF-LAN Lotus Notes channel, or any other services running under the Dispatcher that need to be made aware of the change, be sure to restart such services using the PMDF RESTART utility.

Following are some of the more commonly encountered mm_init errors.

bad equivalence for alias ...
The right hand side of an alias file entry is improperly formatted.

cannot open alias include file...
A file included into the alias file cannot be opened. This typically indicates a protection problem with a file referenced by the file include operator, <. Note that such included files (like the alias file itself) must be world readable.

duplicate alias(es) found ...
Two alias file entries have the same left hand side; you will need to find and eliminate the duplication.

duplicate host in channel table ...
In its literal meaning, this error says that you have two channel definitions in the PMDF configuration that both have the same official host name (line two of the channel definition); see Section 2.3.2. But note that an extraneous blank line in the rewrite rules (upper portion) of your PMDF configuration file causes PMDF to interpret the remainder of the configuration file as channel definitions, and as there are often multiple rewrite rules with the same pattern (left hand side), this then causes PMDF to think it is seeing channel definitions with non-unique official host names. So check your PMDF configuration both for any channel definitions with duplicate official host names, and for any improper blank lines in the upper (rewrite rules) portion of the file.

duplicate mapping name found ...
This error literally means that two mapping tables have the same name, and one of the "duplicates" needs to be removed. However, note that formatting errors in the mapping file can cause PMDF to interpret something not intended as a mapping table name as a mapping table name; for instance, failure to properly indent a mapping table entry will

[2] Note that PMDF stores configuration information in internal tables in memory. To prevent unnecessary use of excessive amounts of virtual memory, these tables are allocated with fixed sizes. The sizes of the tables are controlled by values in the PMDF option file. See Chapter 7 for details on PMDF options.

cause PMDF to think that the left hand side of the entry is actually a mapping table name. So check your mapping file for general format, as well as checking the mapping table names.

error initializing ch_ facility: ...

Note that such errors should not occur in normal operation; only sites that have customized PMDF character set material, or have had installation problems, are likely to encounter such errors. The next two items describe sorts of ch_ facility errors that are simple to resolve; other sorts of ch_ errors, however, often indicate that required PMDF files have not been properly installed or have been unintentionally deleted or otherwise corrupted, and a re-installation of PMDF can be necessary to get the required files properly installed. Contact Process Software if you have questions regarding such an error.

error initializing ch_ facility: compiled character set version mismatch

Such an error generally means that you need to recompile and reinstall your compiled character set tables via the commands:

```
$ PMDF CHBUILD
$ INSTALL REPLACE PMDF_CHARSET_DATA
```

See the documentation for PMDF CHBUILD in Chapter 29 for additional details.

error initializing ch_ facility: no room in ...

Such an error likely means that you need to resize your PMDF character set internal tables and then rebuild the compiled character set tables via the commands

```
$ PMDF CHBUILD/NOIMAGE/MAXIMUM/OPTION
$ PMDF CHBUILD
$ INSTALL REPLACE PMDF_CHARSET_DATA
```

See the documentation for PMDF CHBUILD in Chapter 29 for additional details.

local host alias or proper name too long for system ...

This error literally means that a local host alias or proper name (the optional right hand side in the second or subsequent names in a channel block) is too long. However, note that certain syntax errors earlier in the PMDF configuration file (an extraneous blank line in the rewrite rules, for instance) can cause PMDF to interpret something not intended as a channel definition as a channel definition. So besides checking the indicated line of the configuration file, also check above that line for other syntax errors and in particular, if the line on which PMDF issues this error is intended as a rewrite rule, then be sure to check for extraneous blank lines above it.

mapping name is too long ...

This error literally means that a mapping table name is too long and needs to be shortened. However, note that formatting errors in the mapping file can cause PMDF to interpret something not intended as a mapping table name as a mapping table name; for instance, failure to properly indent a mapping table entry will cause PMDF to think that the left hand side of the entry is actually a mapping table name. So check your mapping file for general format, as well as checking the mapping table names.

no equivalence addresses for alias ...

An entry in the alias file is missing a right hand side (translation value).

no official host name for channel ...

This error indicates that a channel definition block is missing the required second line (the official host name line). See Section 2.3.2 for a discussion of the format of channel definition blocks. In particular, note that a blank line is required *before* and *after* each channel definition block, but a blank line *must not* be present between the channel name and official host name lines of the channel definition; also note that blank lines are *not permitted* in the rewrite rules portion of the PMDF configuration file.

no room in ...

Generally, "no room in" errors are an indication that your current PMDF configuration has not set internal table sizes sufficient for the size of your PMDF configuration, and that it is time to have PMDF resize its internal tables, as described in Section 8.1.4. However, some particular such "no room in ..." error messages can have alternate causes, and such cases are called out below. Any other "no room in" errors not explicitly mentioned are most likely simply an indication of a need to resize internal tables.

no room in channel host table for ...

This error indicates that your configuration's current PMDF internal table sizes are not large enough for the number of host names listed in your channel definitions. However, note that an extraneous blank line in the rewrite rules (upper portion) of your PMDF configuration file causes PMDF to interpret the remainder of the configuration file as channel definitions; with just one such extraneous blank line, PMDF sees just one extra channel but with a lot (all the rest of the rewrite rules) as host names on that channel. So check the line of the file that the error is complaining about—if it is not truly intended as a host name on a channel definition but rather is a line in the rewrite rules section of your configuration file, then check for an extraneous blank line above it.

no room in channel table for ...

This error indicates that your configuration's current PMDF internal table sizes are not large enough for the number of channels defined in your PMDF configuration. See Section 8.1.4.

no room in table for alias ...

This error says that the current PMDF internal table sizes are too small for the number of aliases in the aliases file. This can be resolved either by resizing PMDF's internal table sizes—see Section 8.1.4—or in some cases can be an indication that it would be wise to begin using the PMDF alias database to store some of the aliases currently in the aliases file—see Section 3.1.2.

no room in table for mapping named ...

In its literal meaning, this error says that your configuration's current PMDF internal table sizes are not large enough for your current number of mapping tables. Internal PMDF table sizes can be increased to match your current configuration side—see Section 8.1.4. However, also note that formatting errors in the PMDF mapping file can cause PMDF to think that you have more mapping tables than you really have; for instance, check that mapping table entries are all properly indented.

official host is too long

The official host name for a channel (second line of the channel definition block) is limited to forty characters in length. So if you were trying to use a longer official host name on a channel, shorten it to a "placeholder" name and then use a rewrite rule to match the longer name to the short official host name. Note, however, that certain syntax errors earlier in the PMDF configuration file (an extraneous blank line in the rewrite rules, for instance) can cause PMDF to interpret something not intended as a channel definition as a channel definition; that could result in an intended rewrite rule being interpreted as an official host name. So besides checking the indicated line of the configuration file,

also check above that line for other syntax errors and in particular, if the line on which PMDF issues this error is intended as a rewrite rule, then be sure to check for extraneous blank lines above it.

33.3.2 Compiled Configuration Version Mismatch

One of the functions of the PMDF CNBUILD utility is to compile PMDF configuration information into an image or file that can be loaded quickly.

The compiled format is quite rigidly defined and often changes substantially between different versions of PMDF. Minor changes can also occur as part of mid-version releases.

When such changes occur, an internal version field is also changed so that incompatible formats can be detected. When an incompatible format is detected, PMDF components will halt with a "Compiled configuration version mismatch" error.

The solution to this problem is simply to generate a new compiled configuration with the OpenVMS commands,

```
$ PMDF CNBUILD/OPTION
$ INSTALL REPLACE PMDF_CONFIG_DATA
```

The OpenVMS INSTALL command must be repeated on every cluster node that runs PMDF.

It is also a good idea to use the PMDF RESTART command to restart any resident PMDF server processes so they can obtain updated configuration information.

33.3.3 File Open or Create Errors

In order to send a message, PMDF needs to read configuration files and create message files in the PMDF message queue directories. Configuration files must be readable to the user, which generally implies world read access on the files in the PMDF table directory. During installation, proper protections are assigned to these files. PMDF utilities and procedures which create configuration files also assign proper protections. If the files are reprotected by the system manager or other privileged user or through some site-specific procedure, PMDF can not be able to read configuration information. This will result in "File open" errors or unpredictable behavior. The PMDF TEST/REWRITE utility will report additional information when it encounters problems reading configuration files. See Chapter 29 for information on using this utility.

If PMDF appears to function from privileged accounts but not from unprivileged accounts, then file protections in the PMDF table directory are likely to blame. Check the protections on configuration files and their directories. The only files that should be protected **against** world read access in the table directory are the queue cache database and certain channel specific files such as PhoneNet script files or other channel option files which can contain password information.

"File create" errors usually indicate a problem while creating a message file in a PMDF message queue directory. See Section 33.2.2 for procedures to aid in diagnosing file creation problems. Note that on OpenVMS, PMDF uses a protected shareable image to control the creation of files in the PMDF queue directories. This allows unprivileged users to create such files where they would not normally be allowed access. Make sure that PMDF has been properly started, as the `pmdf_startup.com` procedures installs required known images and defines required logical names in the proper modes. If, for example, the PMDF startup procedure is run by an account with insufficient privileges, an environment can be created which precludes proper operation of the PMDF protected shareable image. In particular, most PMDF logical names must be defined in executive mode.

On OpenVMS, use of the undocumented SET WATCH FILE/CLASS=MAJOR command in DCL can prove an immense help in determining which file or file operation is causing a problem. Use /CLASS=ALL to obtain a verbose amount of information from the XQP; use /CLASS=NOALL to disable the WATCH output.

33.3.4 Illegal Host/domain Errors

Such an error can be returned immediately in response to an address provided to PMDF through a user agent, or the error can be deferred and returned as part of an error return mail message. In all cases, such an error message indicates that PMDF is not able to deliver mail to the specified host. Before diagnosing such problems any further, verify that the address in question is indeed correct and is not misspelled, transcribed incorrectly, or using the name of a host or domain which no longer exists.

Try running the address in question through the PMDF TEST/REWRITE utility. If this utility also returns an "illegal host/domain" error on the address, then PMDF has no rules in its configuration file, `pmdf.cnf` and related files, to handle the address. Verify that you have configured PMDF correctly, that you answered all configuration questions appropriately, and that you have kept your configuration information up to date.

Otherwise, if PMDF TEST/REWRITE does not encounter an error on the address, then PMDF was able to determine how to handle the address, but the network transport would not accept it. You can examine the appropriate log files from the delivery attempt for additional details. Transient network routing or name service errors should not result in returned error messages, though it is possible for badly misconfigured domain name servers to cause such problems.

If you are on the Internet, then check that you have properly configured your TCP/IP channel to support MX record lookups. Many domain addresses are not directly accessible on the Internet and require that your mail system correctly resolve MX entries. If you are on the Internet and you are using a TCP/IP package that PMDF supports for MX lookups, you should have allowed the PMDF configuration utility to enable MX support. If your TCP/IP package is not configured to support MX record lookups, then you will not be able to reach MX-only domains.

33.3.5 Errors in SMTP Channels: os_smtp_* Errors

os_smtp_* errors, *e.g.*, os_smtp_open, os_smtp_read, or os_smtp_write errors, are not PMDF errors *per se*: they correspond to PMDF reporting back about a problem encountered at the network layer. For instance, an os_smtp_open error means that the network connection to the remote side could not be opened, which can be due to addressing errors or channel configuration errors (PMDF configured to attempt to connect to the "wrong" system), but is more commonly due to DNS problems or network connectivity problems (particularly if this is a channel or address that was previously working). os_smtp_read or os_smtp_write errors are usually an indication that the connection was aborted (either by the other side or due to network problems).

Note that network and DNS problems are often transient in nature. It is normal to occasionally see such problems. Indeed, for connections to troublesome systems, it can even be common. So the occasional such error is usually nothing to be concerned about. However, if you are consistently seeing such errors on most messages on a channel, or seeing such errors on most messages to or from a particular remote system, then the errors can be an indication of an underlying network problem.

If you need more information about an os_smtp_* error, enable debugging on the channel in question and get a debug channel log file showing details of the attempted SMTP dialogue; see Section 2.3.4.85. In particular, the timing of exactly when a network problem occurred during the SMTP dialogue tends to be suggestive as to what sort of network or remote side issue might be involved. In some cases, you can also want to do network level debugging (*e.g.*, TCP/IP packet tracing) to see what was sent or received over the wire.

33.3.6 Error Activating Transport IN

If an attempt to send mail using the PMDF foreign protocol prefix in VMS MAIL results in the message

```
%MAIL-E-ERRACTRNS, error activating transport IN
```

then it is likely that PMDF has not been started on the system. Check to see that the system is properly licensed for PMDF and, if so, that the PMDF startup command procedure has been run. Additional message text can accompany the above error indicating problems such as failure to install protected shareable images. If PMDF images are not getting installed as known files, you should check that INSTALL has not run out of GBLPAGES or GBLSECTIONS.

If you are trying to send to PMDF from a non-PMDF node in a cluster and are encountering this error, check whether the OpenVMS `MAIL$SYSTEM_FLAGS` logical has been defined, and specifically been defined to have an odd value (bit 0 set).

33.3.7 No License Error

If the error message

```
%SYSTEM-F-NOLICENSE, operation requires software license
```

is reported when attempting to send mail with PMDF, then the PMDF license is not properly loaded. Either the PMDF license PAKs need to be loaded, or the license needs to be enabled by PMDF. First check to see if the PMDF license is loaded by issuing the command:

```
$ SHOW LICENSE PMDF-product
```

If the PAKs have not been loaded, then load them. After the PAKs are loaded, if the license is for PMDF, then issue a PMDF LICENSE LOAD command.

If the license does shows up properly and if the license is for PMDF, then make sure PMDF's license units have been loaded by issuing the following command:

```
$ PMDF LICENSE LOAD
```

If PMDF reports that you have insufficient license units and you feel that this is incorrect, then make sure that unlicensed cluster nodes are not executing the `pmdf_startup.com` command procedure.

Note that if your license is for PMDF, then PMDF LICENSE LOAD should respond with an informational message either to the effect that the units are loaded, or to the effect that they were already loaded.

33.3.8 Error in qu_init: Usage Level Requires PMDF-MTA Service

If the error message "Usage level requires PMDF-MTA service" is reported by PMDF components, then a PMDF-ACCESS license is loaded on a system that has a PMDF-MTA configuration, that is, a configuration including channels not allowed on a PMDF-ACCESS system.

If this system is supposed to be a PMDF-ACCESS system, then the problem is that the PMDF configuration has non-ACCESS, disallowed components in it. Either manually modify the PMDF configuration file to remove non-ACCESS channels, or use the PMDF CONFIGURE ACCESS utility to generate a new PMDF-ACCESS configuration.

If this system is supposed to be a PMDF-MTA system rather than a PMDF-ACCESS system, then the problem is that a PMDF-ACCESS license has been loaded instead of or in addition to a PMDF-MTA license. Note that the (lesser functionality) PMDF-ACCESS license will override the PMDF-MTA license; you must remove the PMDF-ACCESS license in order for the PMDF-MTA license to take effect. The

```
$ PMDF LICENSE UNLOAD
```

command should be used to ensure that PMDF's own bundle license accounting is not counting this system as a PMDF-ACCESS node. Then the DCL LICENSE command should also be used to ensure that the PMDF-ACCESS license has not been loaded

by OpenVMS' own license accounting; note that the LICENSE MODIFY/EXCLUDE or LICENSE MODIFY/INCLUDE commands are useful in a cluster setup to ensure that the proper licenses are loaded only on the proper nodes.

33.3.9 Line too Long Error on MAIL-11 Channels

There is a limitation in VMS MAIL which prevents a foreign protocol transport, such as PMDF, from sending through VMS MAIL over DECnet to a remote VMS MAIL using block mode I/O. This restriction manifests itself as a problem when PMDF tries to deliver a message which contains very long records through a MAIL_ (MAIL-11) or D (DECnet) channel, or even through the L (local) channel if the recipient address has used MAIL-11 forwarding to forward their messages to another DECnet node.

If you find messages stuck in the message queue for the D channel, check the corresponding log file, *e.g.*, `d_master.log`, for an error of the form

```
$ MAIL/PROTOCOL=PMDF_SHARE_LIBRARY
%RMS-W-RTB, 397 byte record too large for user's buffer
```

The only solution to this problem is to mark your D channel with the `linelength` keyword to force PMDF to soft-wrap (by encoding) lines which exceed 255 characters. For example

```
d logging 733 linelength 255
DECNET-MAIL
```

See Section 2.3.4 for details on the `linelength` keyword.

If you see such an error for messages stuck for the L (local) channel, check whether the recipient has MAIL-11 forwarding set to another DECnet node. If so, encourage the recipient user to use PMDF style forwarding instead – forwarding that will route through PMDF and thence through a D channel marked with the `linelength` keyword as described above. It is also possible to mark the L channel itself with the `linelength` keyword—but that will impact all your L channel users, not just those who forward their messages to other DECnet nodes. By having such users forward through PMDF and a D channel instead, you can restrict the cases where PMDF has to perform (encoding based) soft line wrapping to only those cases where it is strictly required.

33.4 Common Problems and Solutions

The following section lists some of the more common problems encountered during PMDF installation, configuration, and operations.

33.4.1 Changes to Configuration Files or PMDF Databases Do Not Take Effect

If changes to your configuration, mapping, conversion, security, option, or alias files or to PMDF databases do not seem to be taking effect, check to see if you have a compiled configuration, and that you have exited and restarted your mail user agent (*e.g.*, Pine, PMDF MAIL, VMS MAIL, DECwindows MAIL) session, and that you have restarted any PMDF components that need to be made aware of the change. See Section 8.2 for additional discussion.

33.4.2 PMDF Sends Outgoing Mail, but Does Not Receive Incoming Mail

Most PMDF channels depend upon a slave, or server, channel program to receive incoming messages. For most transports supported by PMDF, in particular TCP/IP, you need to make sure that the transport activates the PMDF slave program rather than its standard server.

For PMDF's multithreaded TCP/IP channel this means you need to disable the TCP/IP package's native SMTP server and run the PMDF SMTP server instead; see Section 21.1.3.

If the TCP/IP package is correctly configured to allow the PMDF SMTP server to run, then check that the PMDF SMTP server is indeed running. For the multithreaded SMTP server, this is controlled via the PMDF Dispatcher. The PMDF Dispatcher controls the starting up of an SMTP server or servers, according to your Dispatcher configuration. If the Dispatcher is configured to use a MIN_PROCS value greater than or equal to one for the SMTP service, then there should always be at least one SMTP server process running (and potentially more, according to the MAX_PROCS value for the SMTP service). The DCL command SHOW SYSTEM/NET or the PMDF PROCESS command can be used to check for the presence of SMTP server processes; see Section 33.2.4. Also, the Dispatcher statistics web page can be viewed to check just what connections SMTP processes are currently handling.

33.4.3 POP and IMAP Clients Time Out

The first thing to check is whether the clients are timing out when they try to connect, or when they try to send mail. If the trouble is when users try to connect to read messages, then you need to investigate the POP or IMAP server, according to which sort of client is having difficulties. But note that POP and IMAP clients send mail out using SMTP and the SMTP port, so if the trouble is when users try to connect to send messages, then you need to investigate the SMTP server.

The POP, IMAP, and SMTP servers are controlled by the PMDF Dispatcher. For the server in question, check the Dispatcher configuration for the maximum number of servers allowed for that service (MAX_PROCS), and the maximum number of connections each individual server can handle (MAX_CONNECTIONS). Then check how many such server processes are actually running. If you have more than MAX_PROCS*MAX_ CONNECTIONS users trying to connect simultaneously, then you can want to increase

the total number of servers allowed, or perhaps the number of connections each server can handle—though note that allowing too many connections per server tends to degrade the performance for each particular connection.

Additionally, general system resource and quota problems will of course impact the servers. See, for instance, Section 33.4.4.

33.4.4 Time Outs on Incoming SMTP Connections

Timeouts on incoming SMTP connections are most often related to system resources and the allocation thereof.

Check how many simultaneous incoming SMTP connections you allow. This is controlled by the MAX_PROCS and MAX_CONNECTIONS Dispatcher settings for the SMTP service; the number of simultaneous connections allowed is MAX_PROCS*MAX_CONNECTIONS. If you can afford the system resources, consider raising this number if it is too low for your usage.

Try putting the `slave_debug` keyword on the channels handling incoming SMTP over TCP/IP mail, usually tcp_local. Then look at the resulting `tcp_local_slave.log` log files and try to spot any particular characteristics of the messages that time out. For instance, if incoming messages with large numbers of recipients are timing out, consider using the `expandlimit` keyword on the channel.

Of course, if your system is extremely overloaded and overextended, time outs will be difficult to avoid entirely.

33.4.5 Outgoing TCP/IP Messages Sit in Queue

Errors encountered during TCP/IP delivery are quite often transient in nature and PMDF will generally retain messages when problems are encountered and retry them periodically. It is quite normal on very large networks to experience periodic outages to certain hosts while other host connections work fine. You can examine the log files for errors relating to delivery attempts. You can see error messages such as “Fatal error from smtp_open”. Such errors are not uncommon and are usually associated with a transient network problem. Your TCP/IP package can contain tools such as ping, traceroute, and nslookup to aid in debugging TCP/IP network problems.

One thing to check in your PMDF configuration is that the TCP/IP channel in question has been correctly configured to perform host lookups, according to your needs. If the channel is one that delivers to Internet systems, then it must perform MX lookups using the DNS (nameservers); make sure that your TCP/IP package has been configured to allow MX lookups, and that the TCP/IP channel in question has been marked with a keyword such as `mx` or `randommx`, meaning that MX lookups will be performed. If the TCP/IP channel is one delivering to a private network and you want to use local host table lookups rather than consulting a nameserver, make sure that the channel has

been marked nodns, meaning that local host table lookups will be performed rather than nameserver lookups.

Example 33–1 shows the steps you might use to see why a message is sitting in the queue awaiting delivery to xtel.co.uk. This example assumes you are using Process Software's MultiNet. The basic idea is to duplicate the steps PMDF uses to deliver SMTP mail on TCP/IP.

If you are using some other TCP/IP package, check with your vendor to see what diagnostics tools are available.

Example 33–1 Tracing TCP/IP Mail Delivery

```
$ MULTINET NSLOOKUP/TYPE=MX XTEL.CO.UK ❶

Server:  LOCALHOST
Address:  127.0.0.1

Non-authoritative answer:
XTEL.CO.UK      preference = 10, mail exchanger = nsfnet-relay.ac.uk ❷

$ MULTINET PING NSFNET-RELAY.AC.UK ❸
PING NSFNET-RELAY.AC.UK (128.86.8.6): 56 data bytes
64 bytes from 128.86.8.6: icmp_seq=0 time=490 ms
CANCEL

$ TELNET/PORT=25 NSFNET-RELAY.AC.UK ❹
Trying... [128.86.8.6] %MULTINET-F-ECONNREFUSED, Connection refused
```

❶ First use the NSLOOKUP utility to see what MX records, if any, exist for this host. If no MX records exist, then you should try connecting directly to the host. If MX records do exist, then you must test by connecting to the designated MX relays since PMDF is required to honor MX information preferentially.

❷ In this example, the Domain Name Service returned the name of the designated MX relay for xtel.co.uk. This is the host that PMDF will actually connect to. If more than one MX relay is listed, PMDF would try each in succession.

❸ A simple way to test connectivity to the host is with a PING utility. If no response is received, then you have a network routing or configuration problem. If the problem is on some router over which you have no control, there is not anything you can do except to wait until it is fixed.

❹ If you do have connectivity to the remote host, the next step is to see if it is accepting inbound SMTP connections by using TELNET to the SMTP server port, port 25. If you use TELNET without specifying the port, you can merely discover that the remote host accepts normal TELNET connections. This by no means indicates that it accepts SMTP connections: many systems can accept regular TELNET connections but refuse SMTP connections or vice versa. Thus, you should always do your testing against the SMTP port.

In this example, the remote host is currently refusing connections to the SMTP port. This is undoubtedly why PMDF fails to deliver the message. The connection can

be refused due to a misconfiguration of the remote host or some sort of resource exhaustion, again, on the remote host. There is absolutely nothing you can do locally to solve the problem. Typically, you should just let PMDF continue to retry the message.

If you are running on a TCP/IP network which does not use the Domain Name Service, then you can skip steps ❶ and ❷ and use PING and TELNET directly to the host in question. Be careful to use precisely the host name that PMDF would use, which can be ascertained by examination of the relevant log file from PMDF's last attempt.

Note that if you test connectivity to a TCP/IP host and encounter no problems using interactive tests, it is quite likely that the problem has simply been resolved since PMDF last tried delivering the message. This is not an indication of a problem with PMDF.

33.4.6 PMDF Messages are not Delivered

In addition to message transport problems, there are three other common issues which can lead to messages sitting around unprocessed—or temporarily unprocessed—in the message queues:

1. The message has a low priority. By default, PMDF pays attention to Priority: headers in scheduling message delivery jobs: only messages of normal or urgent Priority: get an immediate delivery attempt, while messages of non-urgent Priority: wait until the next run of the PMDF periodic delivery job.
2. The queue cache database is not synchronized with the messages in the queue directories.

   Message files in the PMDF queue subdirectories which are awaiting delivery are entered into the queue cache database. When channel programs run in order to deliver messages in their queues, they consult the queue cache to determine what messages to process. There are circumstances which can lead to message files in the queue that do not have a corresponding queue cache entry: for example, if the queue cache database has incorrect ownership and protection; see Section 33.2.3. Channel programs will ignore queued messages which do not have a cache entry. On OpenVMS, you can use the DUMP/RECORD command on the queue cache database to check if a particular file is in the queue cache; if it is not, then the queue cache needs to be synchronized.

   The queue cache is normally resynchronized daily. If required, you can manually resynchronize the cache using the OpenVMS command

```
$ PMDF CACHE/SYNCHRONIZE
```

   Once resynchronized, upon the next running of the PMDF periodic delivery job the channel programs should process all messages in their queue.

On OpenVMS, if you are having trouble with the queue cache which is not remedied by a resynchronization, there is an additional command you should use to try to mollify the database:

```
$ @PMDF_COM:convert_cache.com
```

There is a more drastic step, rebuilding the queue cache database, which should only be performed as a last resort, *e.g.*, if disk problems have corrupted your queue cache database, as it will cause loss of some information from the queue cache database. (The sort of information lost includes, but is not limited to, message creation dates, message deferral dates, message expiration dates, and the original message owner information used by the PMDF QM/USER utility to allow users to bounce their own messages.) You can want to consult Process Software before taking this step.

To rebuild the queue cache database, use the OpenVMS commands

```
$ PMDF CACHE/REBUILD
$ PMDF CACHE/CLOSE
$ PMDF CACHE/SYNCHRONIZE
```

3. Channel processing programs fail to run because they cannot create their execution log file (*e.g.,* batch log file).

   Check the access permissions, disk space and quotas, and that there are no version limits set on the PMDF log directory and that none of the files therein have reached a version number of 32,767.

33.4.6.1 Checking Version Limits and Numbers

PMDF log files, when created, are placed in the `PMDF_LOG:` directory. After a period of time which is dependent on the level of PMDF activity on your system, the file version number on one or more log files can reach the RMS version number limit of 32,767. At this point PMDF will be unable to create a new log file and will no longer deliver messages on the associated channel.

PMDF will detect that log file version numbers are getting high and try to shuffle them back down to a safe level. If it is unable to do so, then warning messages will be sent to the postmaster. Certain situations, however, can prevent the warning from getting through. In any case, if you have detected a situation where log file version numbers are getting too high and PMDF has not fixed them for you, you should delete all versions of the log files in question. After that, new logs with the same name will start over from version 1.

Additional version limit problems will occur if version limits are set on the PMDF log directory. Consider the following scenario: A message is enqueued and a delivery job is started, but delivery processing takes an unusually long time due to network congestion. As this first job runs other messages are enqueued and dequeued from the channel, their delivery jobs producing additional log files. Now, if a version limit is ever reached, subsequent jobs will not be able to run because the log file associated with the first job is still open and cannot be purged. The resulting failures in turn lead to significant delivery delays.

The inevitable outcome here is that file version limits cannot be used as a means to control the number of PMDF log files that are created. For this reason, PMDF incorporates facilities to automatically purge accumulated log files back to the limit set by the `PMDF_VERSION_LIMIT` logical. (The default is 5 if this logical is not set.) Version limits are therefore unnecessary and **must not be imposed** on the PMDF log directory.

33.4.7 Message Queues Contain .HELD Files

Mail messages in the PMDF message queue directories generally have a two-digit file extension. If PMDF detects a message that is looping, it will rename the file so that it has an extension of `.HELD`. Message files with such a file extension will not be processed by PMDF channel programs and therefore will not be delivered. This is a safety mechanism to prevent messages from looping indefinitely. Looping messages are detected by having a large number of Received: headers lines.

33.4.7.1 Diagnosing .HELD Files

One possible cause of message loops is user error: a user forwards their messages on system A to system B, and has system B set up to forward back to system A. The solution is for the user to fix their forwarding definitions.

Another common cause of messages loops is PMDF receiving a message that was addressed to your host with a network name that PMDF does not recognize as one of the host's own names. For example, imagine a host which is known to the TCP/IP domain name system and to other hosts and users as example.com, but whose PMDF configuration does not know that. A message is sent to joe@example.com and is accepted by the network and delivered to this host. Since PMDF does not know itself as example.com, it will likely assume that example.com is elsewhere and direct the message back out to the network and unwittingly loop the message back to itself. This loop will continue until PMDF detects the loop and puts the message on hold.

If you detect such a situation, you should try to determine by examination of the message file whether there is a name you should add to your PMDF configuration as a synonym for your official local host name. The Received: lines should show the path the message travels through the loop.

❶ Rewrite rules for first cluster member to official local host.

❷ Rewrite rules for second cluster member to official local host.

❸ We have added this rule so example.com is recognized.

❹ The host name on the L channel (local channel) is always the official local host.

In Example 33–2 we have added example.com as another name for the cluster consisting of milan.example.com and naples.example.com, where the official local host name has been milan.example.com. Mail addressed to joe@example.com will now be properly recognized and locally delivered.

If you do not believe that the name in question should be directed to your host, then you can have to address the problem with a network configuration change or by changing the behavior of a remote mailer.

Example 33–2 `pmdf.cnf` **For milan.example.com**

```
! pmdf.cnf - PMDF configuration file for milan.example.com
! Written by SYSTEM, 19-AUG-2012 21:23
! This file was created by the PMDF configuration generator.
!
! Rewrite rules for the local host/cluster
!
milan                                          $u@milan.example.com ❶
milan.example.com                              $u@milan.example.com
naples                                         $u@milan.example.com ❷
naples.example.com                             $u@milan.example.com
example.com                                    $u@milan.example.com ❸
!
! Rewrite rules for the Internet
                .
                .
                .
l nox_env_to
milan.example.com ❹

                .
                .
                .
```

33.4.7.2 Cleaning Up .HELD Files

After diagnosing and fixing the cause of the loop, `.HELD` files should be renamed to `.00`; *e.g.,* issue the command:

```
$ RENAME PMDF_QUEUE:[tcp_local...]*.HELD *.00
```

Then synchronize the queue cache with the command

```
$ PMDF CACHE/SYNCHRONIZE
```

(Alternatively, the PMDF QM/MAINT utility's RELEASE command can be used to cause message files to cease to be `.HELD` and the PMDF queue cache database to be synchronized.)

Then release the pending "PMDF delivery" job which is holding in the MAIL$BATCH queue. Now, after the resulting jobs have run, look around to see if there are new `.HELD` files. There can very well be some. If there are still `.HELD` files in the original queue directory, then you can not have solved the looping problem. However, you can also find `.HELD` files in other queue directories such as the local channel (L channel) queue directory. This is because PMDF marks a message `.HELD` when it has too many Received: lines in which the local host appears. As the message moved from the original directory to another directory (*i.e.,* moved from one channel to another), PMDF again saw too many Received: lines in the message's header and again marked it `.HELD`. This is to be expected. Simply repeat the process of renaming the `.HELD` files to `.00`, synchronizing the queue cache, and again releasing the pending PMDF delivery job. Repeat this process until there are no more `.HELD` files in any of the channel queue directories.

33.4.8 Messages are Looping

If PMDF detects that a message is looping, that message will be sidelined as a `.HELD` file; see Section 33.4.7 for a discussion. But certain cases can lead to message loops which PMDF can not detect. Some of the more common cases include:

1. A postmaster address is broken.
2. Stripping of Received: headers is preventing PMDF from detecting the message loop.
3. Incorrect handling of notification messages by other mail systems, that are generating reencapsulated messages in response to notification messages.

The first step in dealing with looping messages is to determine why the messages are looping. Useful things to look at are a copy of the problem message file while it is in the PMDF queue area, PMDF mail log entries (if you have the `logging` channel keyword enabled in your PMDF configuration file for the channels in question) relating to the problem message, and PMDF channel debug log files for the channels in question. Determining the From: and To: addresses for the problem message, seeing the Received: headers, and seeing the message structure (type of encapsulation of the message contents), can all help pinpoint which sort of message loop case you are encountering.

For case (1), note that mail systems such as PMDF require that the postmaster address be a functioning address that can receive e-mail. If a message to the postmaster is looping, check that your configuration has a proper postmaster address pointing to an account that can receive messages.

For case (2), note that normal detection of message loops is based on various Received: headers. If Received: headers are being stripped—either explicitly on the PMDF system itself, or more likely on some other system such as a firewall—that interferes with proper detection of message loops. There will likely be two issues to resolve: check that no undesired stripping of Received: headers is occurring so that if a loop does occur it can be short-circuited, and check for the underlying reason why the messages were looping. Possible underlying reasons for the occurrence of the message loop in the first place include: a problem in the assignment of system names or a system not configured to recognize a variant of its own name, a DNS problem, a lack of authoritative addressing information on the system(s) in question, or a user address forwarding error.

For case (3), note that Internet standards require that notification messages (reports of messages being delivered, or messages bouncing) have an empty envelope From: address to prevent message loops. However, some mail systems do not correctly handle such notification messages; such mail systems can, when forwarding or bouncing such a notification message, insert a new envelope From: address of their own. This can then lead to message loops. The solution is to fix the mail system that is incorrectly handling the notification messages.

33.4.9 Received Message is Encoded

Messages sent by PMDF are received in an encoded format; *e.g.,*

```
Date: Sun, 07 Jan 2012 11:59:56 -0500 (EST)
From: "Elvis Presley" <elvis@example.com>
To: priscilla@example.edu
Subject: test message with 8bit data
MIME-Version: 1.0
Content-type: TEXT/PLAIN; CHARSET=ISO-8859-1
Content-transfer-encoding: QUOTED-PRINTABLE

2=00So are the Bo=F6tes Void and the Coal Sack the same?=
```

Such messages appear unencoded when read with a MIME-aware user agent such as PMDF MAIL or PMDF Pine, or when decoded with a decoder such as PMDF DECODE.

The SMTP protocol as set forth by RFC 821 only allows the transmission of ASCII characters. As ASCII is a seven-bit character set, the transmission via SMTP of eight bit characters is illegal. As a practical matter, the transmission of eight bit characters over SMTP is known to cause a variety of problems with some SMTP servers (*e.g.,* cause SMTP servers to go into compute bound loops, cause mail messages to be sent over and over again, crash SMTP servers, wreak havoc with user agents or mailboxes which cannot handle eight bit data, *etc.*).

Until the advent of RFC 1425 and RFC 1426, an SMTP client had only three alternatives when presented with a message containing eight bit data: return the message to the sender as undeliverable, encode the message, or send it anyhow in direct violation of RFC 821. None of these alternatives were pleasant; prior to version 4.2, PMDF chose the latter of the three owing to the lack of a standardized encoding format. However, with the recent advent of MIME (first specified in RFCs 1521 and 1522, and updated in RFCs 2045–2049) and the SMTP extensions work (RFC 1425 and RFC 1426), there are now standard encodings which can be used to encode eight bit data using the ASCII character set and mechanisms to negotiate, between the SMTP client and server, whether or not eight bit data will be accepted as is by the server without first being encoded.

When recipients receive encoded messages such as those shown above with a MIME content type of TEXT/PLAIN, then invariably the original message contained eight bit characters and the remote SMTP server to which the PMDF SMTP client transferred the message did not support the transfer of eight bit data. PMDF then had to encode the message.

Users can avoid sending messages with eight bit characters through use of PMDF MAIL's `/EIGHTBIT` qualifier to the `SEND`, `FORWARD`, and `REPLY` command. See the OpenVMS Edition of the *PMDF User's Guide* for details.

33.4.10 From: Address Missing in Notifications from PMDF

Occassionally users or postmasters on other mail systems will complain that PMDF is losing, dropping, forgetting, or otherwise omitting the envelope From: address in messages it sends. You can be presented with a message header fragment like the one shown below

```
From   Thu Jul 11 11:50:23 2012
Received: from vulcan.ajax.com by monster.ajax,com via SMTP
  (930416.SGI/931108.SGI.ANONFTP) for xxxx id AA21154;
  Thu, 11 Jul 02 11:50:23 +1100
Date: Thu, 11 Jul 2012 11:49:26 +1000
From: PMDF Mail Server <postmaster@vulcan.ajax.com>
```

Note how in the first line there is a noticeable blank space between the "From" and date? This header line is often referred to as the "colonless From line" and it gives the envelope From: address for the message. That blank space indicates that the message had no envelope `From:` address; that is, it had what is called in the mail business a "null return path". Note further that this was an automatically generated mail message as suggested by the RFC 822 `From:` address of postmaster@vulcan.ajax.com.

The relevant standards require that automatically generated messages such as non-delivery notifications and delivery receipts use a null return path. As mailers are supposed to bounce mail to the envelope `From:` address, this helps to prevent mail loops from occurring. [4]

If someone complains about the missing `From:` address, ask them to send you a sample offending message. Determine if it was an automatically generated message. If it was, then explain to them that if their mailer or user agent is incapable of handling null return paths then it is incompliant with RFC 821 and 1123. Refer them to Paragraph 8 of Section 3.6 and the second paragraph of the MAIL command description in Section 4.1.1 in RFC 821. Further point out that were you to change your mailer to use a non-null return path for automatically generated notifications, then you would be violating the Internet Host Requirements; specifically, you would be in violation of Section 5.3.3 of RFC 1123.

Now, if for some reason you absolutely must generate non-null return paths in your notification messages, then you can do so with the RETURN_ENVELOPE option of the PMDF option file; see Section 7.3.4. Or to generate non-null return paths in notification messages only for a particular channel or channels, you can use the `returnenvelope` channel keyword; see Section 2.3.4.64. Be warned: Use of either the option or the channel keyword will put you in violation of the Internet Host Requirements and, more importantly, can lead to looping mail. Looping mail will not only inconvenience you but can cause serious problems for some unfortunate site which gets into a loop with your system. Also, keep in mind that changing PMDF's behavior so as not to cause problems for a broken mailer which cannot handle null return paths does not really fix anything: Other mailers over which you have no control will continue to send the broken mailer messages with null return paths. The only satisfactory solution in this situation is to fix the broken mailer.

[4] Some mailers will preferentially send notifications to the address specified with the non-standard `Errors-to:` or `Warnings-to:` header lines. By default, PMDF itself sends notifications to the envelope `From:` address, unless configured otherwise via the `USE_ERRORS_TO` and `USE_WARNINGS_TO` PMDF options.

33.4.11 MF, MB, or other $M... Files

When using PMDF-MR, sometimes files with names of the form `$Mx$...`, *e.g.*,

```
$MF$01H868YPNYBMAW31.00;1
$MB$01H798YCNYASAW41.00;1
```

are found in the PMDF queue directory. These are temporary files created by Message Router and the HP MRIF interface. You need to check the PMDF-MR log files, usually `mr_local_master.log` or `mrif_*_slave.log` in the PMDF log directory, to see if there is a problem with processing messages to or from Message Router. There are also some pathological cases where Message Router leaves files behind, but these are rare. If no errors are found in the log files, these files can be deleted. You should monitor the queue directory and see if they reappear.

33.4.12 @DIS Mailing List Expands into the Message Header

The only solution to the problem of @DIS mailing list addresses expanding into the To: or other header is to stop using VMS MAIL `@DIS` distribution lists or `@DIS` style addresses within PMDF MAIL.

PMDF has absolutely no control over the expansion of VMS MAIL `@DIS` mailing lists. The expansion is done, and necessarily so, by VMS MAIL. It cannot work any other way: VMS MAIL must look at each address from the list, determine if it is a pure MAIL-11 address, a DECnet MAIL-11 address, or an address involving a foreign protocol interface, and dispatch it accordingly. PMDF is handed, one address at a time, those addresses from the list which are intended for PMDF. (Any addresses that are not intended for PMDF, VMS MAIL does not even tell PMDF about; see Section 33.4.13.) PMDF does not know that a mailing list was used. While PMDF is passed a pure text string containing the exact (original) To: address presented by the user (*e.g.,* `@STAFF.DIS`), the context under which to evaluate that string does not exist. It can have been entered on a remote DECnet node in which case any incomplete file specifications cannot be resolved, and similarly for addresses effected through process logical names. Moreover, the string is just plain text and there are no assurances that it contains anything meaningful (*e.g.,* TM%"bob\[123,456]").

To prevent @DIS mailing lists from being expanded into the message headers, stop using `@DIS` style addresses and begin using PMDF's mailing list facilities. See Section 4.1 for information on creating system-wide mailing lists. Direct your users to the OpenVMS Edition of the *PMDF User's Guide* for information on setting up their own mailing lists.

33.4.13 Some Addresses Missing from Headers of Messages Sent from VMS MAIL

VMS MAIL only hands over to PMDF those addresses that have the IN%"..." wrapper around them (or any synonyms for IN% you have defined on your system); VMS MAIL does not even tell PMDF about any purely VMS MAIL or DECnet MAIL or other foreign protocol addresses. PMDF of course can only construct headers showing the addresses VMS MAIL told PMDF about. Therefore if a VMS MAIL user sends a message to multiple recipients where some of the recipients are *not* handled via PMDF, then the recipients of the messages that went through PMDF will not see those other, non-PMDF recipients.

The only solution is to ensure that all addresses go through PMDF, by using the IN%"..." wrapper around all such addresses, or by using PMDF MAIL.

33.4.14 From: Addresses Contain SMTP%, EDU%, or Other Prefixes

When PMDF is correctly installed, all inbound mail from any network protocol and transport supported by PMDF will be delivered to VMS MAIL users with the PMDF foreign protocol prefix, “IN%”, placed at the beginning of the From: address. This ensures complete repliability, back through PMDF, of messages. If messages are coming in with any other foreign protocol prefix, then messages can not be properly repliable, and additional PMDF functionality, such as the DELIVER facility, will not work.

Incoming mail with a prefix other than “IN%” usually indicates that some PMDF slave or server program has not been properly installed or has encountered some sort of error. For TCP/IP channels, verify that you have disabled the TCP/IP package’s own native SMTP server, and are using the PMDF SMTP server instead; see Chapter 21.

33.4.15 VMS MAIL Exits or Hangs

There is a bug in VMS MAIL which is manifested when a foreign protocol address is entered at the To: or CC: prompts and the trailing double quote character, ", is omitted. For example

```
MAIL> send
To:     in%"user@domain
%LIB-F-SYNTAXERR, string syntax error detected by LIB$TPARSE
$
```

Under older versions of OpenVMS, MAIL will go into an infinite loop rather than exit with the fatal error. This problem has nothing to do with PMDF and cannot be avoided; indeed, it occurs before PMDF is ever invoked by VMS MAIL.

33.5 Contacting Process Software Technical Support

Process Software provides technical support only for sites with a current maintenance agreement.

If you obtained PMDF from an authorized Process Software distributor, then technical support in your timezone can be most efficiently obtained from your distributor. You can also contact Process Software directly if you want.

Process Software technical support can be contacted at:

Process Software, LLC
959 Concord Street
Framingham, MA 01701 USA
+1 508 879 6994
+1 508 879 0042 (FAX)
support@process.com

The public mailing list for PMDF sites, info-pmdf@process.com, is another useful source of input and advice from other sites using PMDF.

When reporting a question or problem to Process Software technical support:

- Please include the output of the PMDF VERSION command. This will include the architecture type for your system, the operating system version for your system, the PMDF installed version number, and the version of your PMDF shared library image.
- If the question regards a message, please include an *entire* sample message, in particular including all message headers. When sending a sample message via e-mail, please extract the sample message (with all headers) to a file and then send that entire file as an attachment; this is much preferable to forwarding the message as a message since forwarding as a message leaves headers (which can be an essential clue) subject to further processing.
- If the question regards the operation of a particular channel, please include a debug channel log file; see Section 2.3.4.85 for a discussion of obtaining channel debug log files.
- Be prepared to send copies of relevant PMDF configuration files, if Process Software technical support requests them.

34 Maintenance and Troubleshooting on UNIX

Under normal conditions PMDF accepts, processes, and promptly delivers messages. If a message can not be delivered immediately, PMDF will periodically retry the delivery. Intervention by a system or network manager is generally not required, as PMDF is designed to perform its functions unattended as much as possible. It is a good idea, however, to periodically inspect the PMDF queue directories. System or network problems over which PMDF has no control can adversely affect normal operation. In addition, unusual behavior from PMDF can often point out network problems which can otherwise go unnoticed.

See also Chapter 31, which discusses monitoring PMDF activity. A baseline for what normal PMDF activity looks like at your site can be helpful when questions arise.

34.1 Background on PMDF Operation

In order to diagnose and repair PMDF message delivery problems, you need some understanding of the steps that PMDF goes through to process and deliver a message. Each time a message is handed to PMDF, a file is created in the appropriate message queue subdirectory. Each channel has a corresponding subdirectory. The subdirectory into which the message is placed is that of the destination channel which will handle the message next. For example, a message coming in from any source to be delivered to a local user will be placed in the subdirectory corresponding to the local, or L, channel. A message coming in from UUCP to be routed on through SMTP over TCP will be placed in the subdirectory for the outbound TCP/IP channel, such as tcp_local.

Once the message is stored in the appropriate queue subdirectory, a PMDF processing job is usually submitted to the PMDF Job Controller. This step is omitted for channels marked with the `periodic` keyword. The processing job will attempt to deliver the message. If it cannot be delivered and the failure is judged to be a permanent condition, the message is returned to the sender and the local postmaster. If the failure is a temporary error condition, the message is left in the message queue to be retried later.

For additional discussion of PMDF operation, see also Chapter 1.

34.2 Standard Maintenance Procedures

In tracking down problems with PMDF you should first determine whether the problem occurs before or after a message is entered into the message queue; then such messages were never accepted: configuration problems, or environmental problems (*e.g.*, disk space or quota problems), or the absence of PMDF servers (such as the Dispatcher and its services) can prevent PMDF from accepting messages; or network connectivity or routing problems could mean that the messages are stuck or misrouted on a remote

system. If messages do not get placed in a queue directory at all, or get put into the wrong queue directory, you probably have a configuration problem.

To track an errant mail message, you can start by examining each step of the process for errors. The subsections below discuss investigating these processing steps.

34.2.1 Check the PMDF Configuration

Use the `pmdf test -rewrite` utility, described in Chapter 30, to test the response of your configuration to addresses. Certain basic sorts of problems in the PMDF configuration, such as clear syntax errors in the PMDF configuration, will cause the utility to issue an error message. Otherwise, the utility will show address rewrites that will be applied as well as the channel to which messages would ultimately be queued. If the output is not what you expect you can need to modify your configuration.

34.2.2 Check Message Queue Directories

Check whether messages are present under the PMDF message queue directory, `/pmdf/queue/`. Shell commands such as `ls` or the `pmdf qm -maint` utility's `directory` command can be used to check for the presence of expected message files under the PMDF message queue directory.

If the `pmdf test -rewrite` output looks correct, check that messages are actually being placed in the PMDF message queue subdirectories. If not, you can have a problem with file space on that disk.

34.2.3 Check the Ownership of Critical Files

You should have a `pmdf` account, created before you installed PMDF. The directories `/pmdf/queue`, `/pmdf/log`, and `/pmdf/table/queue_cache`, and all subdirectories and files under them should be owned by the `pmdf` account. The `/pmdf/tmp` directory should also be owned by the `pmdf` account. If the protection and ownership are not correct for the queue cache database, messages can not be entered into the queue cache, and the queue cache will become desynchronized. Commands such as the following can be used to check the protection and ownership of these directories:

```
# ls -l -p -d /pmdf/queue
drwx------   6 pmdf     bin          512 Feb  7 09:32 /pmdf/queue/
# ls -l -p -d /pmdf/log
drwx------   2 pmdf     bin         1536 Mar 10 20:00 /pmdf/log/
# ls -l -p -d /pmdf/table/queue_cache
drwx------   2 pmdf     bin          512 Mar 10 15:03 /pmdf/table/queue_cache/
# ls -l -p -d /pmdf/tmp
drwx------   2 pmdf     bin          512 Feb  7 10:00 /pmdf/tmp/
```

Then check that any files and subdirectories of /pmdf/queue and /pmdf/log are owned by the pmdf account using commands such as:

```
# ls -l -p -R /pmdf/queue
# ls -l -p -R /pmdf/log
```

34.2.4 Checking that the Job Controller and Dispatcher are Present

The PMDF Job Controller handles the execution of PMDF processing jobs, including most outgoing (master) channel jobs.

Some PMDF channels, such as PMDF's multithreaded TCP/IP channels or Lotus Notes channels, include resident server processes that process incoming messages—such servers handle the slave (incoming) direction for the channel. In addition, remote clients using protocols such as POP and IMAP download their messages by connecting to PMDF's POP or IMAP servers.

The PMDF Dispatcher handles the creation of such PMDF servers. Dispatcher configuration options control whether such servers are available at all, and if available, how many such servers are created and when, and how many connections each server can handle. Most sites, for instance, will choose to have the Dispatcher always keep at least one SMTP server process resident.

The command

```
# pmdf process
```

can be used to check that the PMDF Job Controller and PMDF Service Dispatcher are present, and to see if there are PMDF servers and processing jobs running. Under idle conditions the command should result in output similar to that seen in Figure 34–1 or Figure 34–2.

Figure 34–1 Basic Output of pmdf process

```
# pmdf process
pmdf     13421 IW   Nov 17       0:00.09 /pmdf/bin/dispatcher ❶
pmdf     13427 IW   Nov 17       0:00.24 /pmdf/bin/job_controller ❷
```

Figure 34–2 Output of pmdf process With Optional Processes

```
# pmdf process
pmdf     26031 I      Nov 17     0:01.25 <DISPATCHER> ❶
pmdf     26110 IW     Nov 17     0:00.17 /pmdf/bin/job_controller ❷
pmdf     26128 IW     Nov 17     0:00.06 /pmdf/bin/x400_tsapd -r ❸
pmdf     23217 S      10:00:25   0:00.00 <LNSLAVE> ❹
pmdf     23218 S      10:00:26   0:00.00 <LNMASTER>
pmdf     23222 S      10:03:19   0:29.00 <SMTP> ❺
pmdf     23225 S      10:03:19   0:29.00 <POP3-01> ❻
pmdf     23305 S      10:03:19   0:29.00 <IMAP-01> ❼
```

The Job Controller and Dispatcher jobs, indicated ❶ and ❷ in the figures, should always be present during normal operation of PMDF. Additional jobs can be present as well, if your system is currently processing messages.

❶ The PMDF Dispatcher should always be present. It is responsible for creating PMDF server processes. The PMDF Dispatcher process name is "/pmdf/bin/dispatcher" when the Dispatcher is first created; if the Dispatcher has been restarted, the name will be "<DISPATCHER>".

❷ The PMDF Job Controller should always be present. It is responsible for handling PMDF channel jobs.

❸ Sites using PMDF-X400 should have a TSAPD process present. It is responsible for listening for incoming X.400 connections.

❹ Sites using PMDF-LAN Lotus Notes channels should have one LN master and one LN slave process present per LN channel.

❺ Sites would normally have at least one SMTP server process present and possibly more, depending on the number of current SMTP connections.

❻ Sites using the PMDF POP3 server would normally have at least one POP3 server process present and possibly more, depending on the number of current POP3 connections.

❼ Sites using the PMDF IMAP server would normally have at least one IMAP server process present and possibly more, depending on the number of current IMAP connections.

If the Job Controller or Dispatcher is not present, you should issue the command:

```
# pmdf startup
```

Also, the Dispatcher's currently active connections can be monitored from a web browser; see Section 11.7.

In addition to the sorts of processes described above that should always be present, the Job Controller should create transient channel processing jobs that process a message or a few messages and complete their work. The `pmdf process` command will include any such current channel processing jobs in its output. Figure 34–3 shows an example of `pmdf process` output that includes actively executing channel jobs.

Figure 34–3 Output of `pmdf process` With Channel Jobs

```
# pmdf process
    pmdf  7945 S   May_26        0:08 imapd
    pmdf 19072 S 07:24:53        0:05 /pmdf/bin/job_controller
    pmdf  3885 S 13:52:40        0:12 <SMTP>
    pmdf  9452 S 18:15:18        0:00 imapd
    pmdf  2047 S   Jun_02        0:04 /pmdf/bin/dispatcher
    pmdf  9664 S 18:18:55        0:00 /pmdf/bin/tcp_smtp_client ❶
    pmdf  9932 R 18:22:56        0:03 /pmdf/bin/l_master ❷
```

❶ An outbound TCP/IP channel job is running.

❷ The local channel is running.

34.2.5 Check Processing Log Files

If PMDF processing jobs run properly but the message stays in the message queue directory, you can examine the log files to see what is happening. All log files are created in the directory `/pmdf/log` . Log file name formats for various PMDF processing jobs are shown in Table 34–1.

Table 34–1 PMDF Log Files on UNIX

	File name	Log file contains
	*channel*_master.log–*uniqueid*	Output of master program (usually client) for *channel*
	*channel*_slave.log–*uniqueid*	Output of slave program (usually server) for *channel*
	dispatcher.log–*uniqueid*	Dispatcher logging, if the Dispatcher DEBUG option has been set
	job_controller.log–*uniqueid*	Job controller logging, if the Job Controller option DEBUG=1 has been set
	*server-name*_server.log–*uniqueid*	Logging for server *server-name*
	*server-name*_thread.log–*uniqueid*	Per thread errors for *server-name*
†	post.log–*uniqueid*	Log output for the periodic PMDF delivery job
†	return.log–*uniqueid*	Log output for the periodic PMDF message bouncer job

†The name and existence of such a log file is specified when the job is initially scheduled by `cron`; Process Software recommends using `crontab` entries that specify a log file of the name shown.

Channel log files are not created unless an error occurs or unless debugging output is enabled for the channel with the `master_debug` channel keyword or `slave_debug` channel keyword. See Section 2.3.4.85 for more information.

Each new log file is created with a unique id to avoid overwriting an earlier log written by the same channel. You can use the `pmdf find` utility to aid in finding the desired "version" of a log file. You can purge back older log files using the `pmdf purge` command.

34.2.6 Running a Channel Program Manually

While diagnosing a PMDF delivery problem it can be useful to run a PMDF delivery job by hand, particularly after enabling debugging output for one or more channels. The command

```
# pmdf submit channel-name
```

will notify the PMDF Job Controller to run the channel. If debugging is enabled for the channel in question, `pmdf submit` will create a log file in `/pmdf/log` as described in Table 34–1.

The command

```
# pmdf run channel-name
```

will perform outbound delivery for the channel *channel-name* under the currently active process, with output directed to your terminal. This can be more convenient than submitting a job, particularly if you suspect problems with job submission itself.

34.3 General Error Messages

There are a number of general sorts of issues that can interfere with the running of a variety of PMDF components; such issues include general syntax errors in a PMDF configuration, or license problems, or disk/quota problems leading to trouble writing files. The following sections describe some of the more common general error messages.

Note that the `pmdf test -rewrite` utility will give warnings of many such common issues and with more detailed error messages than some other components of PMDF can display. So `pmdf test -rewrite` can be a useful diagnostic tool: see if it is getting an error and if so, what.

Specific PMDF components can also issue other error messages, specific to that component. So when you an encounter an error not described below, see also the documentation on the PMDF component in question.

34.3.1 Errors in mm_init, such as "No room in ..." Errors

An "Error in mm_init" error generally indicates a PMDF configuration problem. Thus the `pmdf test -rewrite` utility, which is often used to test the health of a PMDF configuration, can return such an error, as can other utilities such as `pmdf cnbuild`, or a channel, or server, or user agent trying to run can return such an error.

In particular, one of the more commonly encountered sorts of mm_init errors is a "No room in table" error or similar "No room in ..." sort of error. Generally, "no room in" errors are an indication that your current PMDF configuration has not set internal table sizes sufficient for the size of your PMDF configuration, and that it is time to have PMDF resize its internal tables, as described in Section 8.1.4.[1] However, some particular such "no room in ..." error messages can have alternate causes, and such cases are called out below. Any other "no room in" errors not explicitly mentioned are most likely simply an indication of a need to resize internal tables.

[1] Note that PMDF stores configuration information in internal tables in memory. To prevent unnecessary use of excessive amounts of virtual memory, these tables are allocated with fixed sizes. The sizes of the tables are controlled by values in the PMDF option file. See Chapter 7 for details on PMDF options.

Rather than manually calculating and setting table sizes, you should use the `pmdf cnbuild` utility to automatically resize the tables for you. See Section 8.1.4 for exact instructions on doing this.

If you use a compiled configuration, then be sure to recompile your configuration and reinstall it after resizing the tables.

If you're using the PMDF multithreaded SMTP server or the PMDF-LAN Lotus Notes channel, or any other services running under the Dispatcher that need to be made aware of the change, be sure to restart such services using the `pmdf restart` utility.

Following are some of the more commonly encountered mm_init errors.

bad equivalence for alias ...
The right hand side of an alias file entry is improperly formatted.

cannot open alias include file...
A file included into the alias file cannot be opened. This typically indicates a protection problem with a file referenced by the file include operator, <. Note that such included files (like the alias file itself) must be world readable.

duplicate alias(es) found ...
Two alias file entries have the same left hand side; you will need to find and eliminate the duplication.

duplicate host in channel table ...
In its literal meaning, this error says that you have two channel definitions in the PMDF configuration that both have the same official host name (line two of the channel definition); see Section 2.3.2. But note that an extraneous blank line in the rewrite rules (upper portion) of your PMDF configuration file causes PMDF to interpret the remainder of the configuration file as channel definitions, and as there are often multiple rewrite rules with the same pattern (left hand side), this then causes PMDF to think it is seeing channel definitions with non-unique official host names. So check your PMDF configuration both for any channel definitions with duplicate official host names, and for any improper blank lines in the upper (rewrite rules) portion of the file.

duplicate mapping name found ...
This error literally means that two mapping tables have the same name, and one of the "duplicates" needs to be removed. However, note that formatting errors in the mapping file can cause PMDF to interpret something not intended as a mapping table name as a mapping table name; for instance, failure to properly indent a mapping table entry will cause PMDF to think that the left hand side of the entry is actually a mapping table name. So check your mapping file for general format, as well as checking the mapping table names.

error initializing ch_ facility: ...
Note that such errors should not occur in normal operation; only sites that have customized PMDF character set material, or have had installation problems, are likely to encounter such errors. The next two items describe sorts of ch_ facility errors that are simple to resolve; other sorts of ch_ errors, however, often indicate that required PMDF files have not been properly installed or have been unintentionally deleted or otherwise corrupted, and a re-installation of PMDF can be necessary to get the required files properly installed. Contact Process Software if you have questions regarding such an error.

error initializing ch_ facility: compiled character set version mismatch

Such an error generally means that you need to recompile and reinstall your compiled character set tables via the command:

```
# pmdf chbuild
```

See the documentation for `pmdf chbuild` in Chapter 30 for additional details.

error initializing ch_ facility: no room in ...

Such an error likely means that you need to resize your PMDF character set internal tables and then rebuild the compiled character set tables via the commands

```
# pmdf chbuild -noimage -maximum -option
# pmdf chbuild
```

See the documentation for `pmdf chbuild` in Chapter 30 for additional details.

local host alias or proper name too long for system ...

This error literally means that a local host alias or proper name (the optional right hand side in the second or subsequent names in a channel block) is too long. However, note that certain syntax errors earlier in the PMDF configuration file (an extraneous blank line in the rewrite rules, for instance) can cause PMDF to interpret something not intended as a channel definition as a channel definition. So besides checking the indicated line of the configuration file, also check above that line for other syntax errors and in particular, if the line on which PMDF issues this error is intended as a rewrite rule, then be sure to check for extraneous blank lines above it.

mapping name is too long ...

This error literally means that a mapping table name is too long and needs to be shortened. However, note that formatting errors in the mapping file can cause PMDF to interpret something not intended as a mapping table name as a mapping table name; for instance, failure to properly indent a mapping table entry will cause PMDF to think that the left hand side of the entry is actually a mapping table name. So check your mapping file for general format, as well as checking the mapping table names.

no equivalence addresses for alias ...

An entry in the alias file is missing a right hand side (translation value).

no official host name for channel ...

This error indicates that a channel definition block is missing the required second line (the official host name line). See Section 2.3.2 for a discussion of the format of channel definition blocks. In particular, note that a blank line is required *before* and *after* each channel definition block, but a blank line *must not* be present between the channel name and official host name lines of the channel definition; also note that blank lines are *not permitted* in the rewrite rules portion of the PMDF configuration file.

no room in ...

Generally, “no room in” errors are an indication that your current PMDF configuration has not set internal table sizes sufficient for the size of your PMDF configuration, and that it is time to have PMDF resize its internal tables, as described in Section 8.1.4. However, some particular such “no room in ...” error messages can have alternate causes, and such cases are called out below. Any other “no room in” errors not explicitly mentioned are most likely simply an indication of a need to resize internal tables.

no room in channel host table for ...

This error indicates that your configuration's current PMDF internal table sizes are not large enough for the number of host names listed in your channel definitions. However, note that an extraneous blank line in the rewrite rules (upper portion) of your PMDF configuration file causes PMDF to interpret the remainder of the configuration file as channel definitions; with just one such extraneous blank line, PMDF sees just one extra channel but with a lot (all the rest of the rewrite rules) as host names on that channel. So check the line of the file that the error is complaining about—if it is not truly intended as a host name on a channel definition but rather is a line in the rewrite rules section of your configuration file, then check for an extraneous blank line above it.

no room in channel table for ...

This error indicates that your configuration's current PMDF internal table sizes are not large enough for the number of channels defined in your PMDF configuration. See Section 8.1.4.

no room in table for alias ...

This error says that the current PMDF internal table sizes are too small for the number of aliases in the aliases file. This can be resolved either by resizing PMDF's internal table sizes—see Section 8.1.4—or in some cases can be an indication that it would be wise to begin using the PMDF alias database to store some of the aliases currently in the aliases file—see Section 3.1.2.

no room in table for mapping named ...

In its literal meaning, this error says that your configuration's current PMDF internal table sizes are not large enough for your current number of mapping tables. Internal PMDF table sizes can be increased to match your current configuration side—see Section 8.1.4. However, also note that formatting errors in the PMDF mapping file can cause PMDF to think that you have more mapping tables than you really have; for instance, check that mapping table entries are all properly indented.

official host is too long

The official host name for a channel (second line of the channel definition block) is limited to forty characters in length. So if you were trying to use a longer official host name on a channel, shorten it to a "placeholder" name and then use a rewrite rule to match the longer name to the short official host name. Note, however, that certain syntax errors earlier in the PMDF configuration file (an extraneous blank line in the rewrite rules, for instance) can cause PMDF to interpret something not intended as a channel definition as a channel definition; that could result in an intended rewrite rule being interpreted as an official host name. So besides checking the indicated line of the configuration file, also check above that line for other syntax errors and in particular, if the line on which PMDF issues this error is intended as a rewrite rule, then be sure to check for extraneous blank lines above it.

34.3.2 Compiled Configuration Version Mismatch

One of the functions of the `pmdf cnbuild` utility is to compile PMDF configuration information into an image or file that can be loaded quickly.

The compiled format is quite rigidly defined and often changes substantially between different versions of PMDF. Minor changes can also occur as part of mid-version releases.

When such changes occur an internal version field is also changed so that incompatible formats can be detected. When an incompatible format is detected PMDF components will halt with a "Compiled configuration version mismatch" error.

The solution to this problem is simply to generate a new compiled configuration with the UNIX command,

```
# pmdf cnbuild -option
```

It is also a good idea to use the `pmdf restart` command to restart any resident PMDF server processes so they can obtain updated configuration information.

34.3.3 Swap Space Errors

For proper PMDF operation it is important to have your system configured with enough swap space. How much swap space will be required will depend upon what components of PMDF you are using and how heavily they are used, as well as on what other non PMDF programs are running on the system. Regarding PMDF components, for instance, heavy POP or IMAP usage will increase the swap space needed. Note that a typical general system tuning recommendation, regardless of PMDF, is to have swap space at least three times the amount of main memory.

For PMDF, at a minimum you should have at least 320 megabytes of swap space configured on Solaris SPARC or at least 250 megabytes on Solaris x86. On a PMDF system with more than minimal usage, better values would be to start with at least 750 megabytes of swap space on Solaris SPARC or at least 500 megabytes on Solaris x86.

In particular, errors such as

```
pmdf: fatal: /pmdf/lib//libpmdf.so: can't set protections on segment: errno=11
```

from various PMDF components, or in the PMDF Job Controller log file an error such as

```
jbc_channels: chan_execute [1]: fork failed: Not enough space
```

are typical symptoms of a lack of swap space.

Note that shell commands such as `swap -s` and, at the time when PMDF processes are busy, `ps -elf` can be useful in seeing how much swap space you have available and used.

34.3.4 File Open or Create Errors

In order to send a message, PMDF needs to read configuration files and create message files in the PMDF message queue directories. Configuration files must be readable to the user, which generally implies world read access on the files in the PMDF table directory. During installation, proper protections are assigned to these files. PMDF utilities and procedures which create configuration files also assign proper protections. If the files are reprotected by the system manager or other privileged user or through some site-specific procedure, PMDF can not be able to read configuration information. This will result in "File open" errors or unpredictable behavior. The `pmdf test -rewrite` utility will report additional information when it encounters problems reading configuration files. See Chapter 30 for information on using this utility.

If PMDF appears to function from privileged accounts but not from unprivileged accounts, then file protections in the PMDF table directory are likely to blame. Check the protections on configuration files and their directories. The only files that should be protected **against** world read access in the table directory are the queue cache database and PhoneNet script files or other channel option files which can contain password information.

"File create" errors usually indicate a problem while creating a message file in a PMDF message queue directory. See Section 34.2.2 for procedures to aid in diagnosing file creation problems.

34.3.5 Illegal Host/domain Errors

Such an error can be returned immediately in response to an address provided to PMDF through a user agent, or the error can be deferred and returned as part of an error return mail message. In all cases, such an error message indicates that PMDF is not able to deliver mail to the specified host. Before diagnosing such problems any further, verify that the address in question is indeed correct and is not misspelled, transcribed incorrectly, or using the name of a host or domain which no longer exists.

Try running the address in question through the `pmdf test -rewrite` utility. If this utility also returns an "illegal host/domain" error on the address, then PMDF has no rules in its configuration file, `pmdf.cnf` and related files, to handle the address. Verify that you have configured PMDF correctly, that you answered all configuration questions appropriately, and that you have kept your configuration information up to date.

Otherwise, if `pmdf test -rewrite` does not encounter an error on the address, then PMDF was able to determine how to handle the address, but the network transport would not accept it. You can examine the appropriate log files from the delivery attempt for additional details. Transient network routing or name service errors should not result in returned error messages, though it is possible for badly misconfigured domain name servers to cause such problems.

If you are on the Internet then check that you have properly configured your TCP/IP channel to support MX record lookups. Many domain addresses are not directly accessible on the Internet and require that your mail system correctly resolve MX entries. If you are on the Internet and your TCP/IP is configured to support MX records, you should

have allowed the PMDF configuration utility to enable MX support; see Chapter 21. If your TCP/IP package is not configured to support MX record lookups, then you will not be able to reach MX-only domains.

34.3.6 Errors in SMTP Channels: os_smtp_* Errors

os_smtp_* errors, *e.g.*, os_smtp_open, os_smtp_read, or os_smtp_write errors, are not PMDF errors *per se*: they correspond to PMDF reporting back about a problem encountered at the network layer. For instance, an os_smtp_open error means that the network connection to the remote side could not be opened, which can be due to addressing errors or channel configuration errors (PMDF configured to attempt to connect to the "wrong" system), but is more commonly due to DNS problems or network connectivity problems (particularly if this is a channel or address that was previously working). os_smtp_read or os_smtp_write errors are usually an indication that the connection was aborted (either by the other side or due to network problems).

Note that network and DNS problems are often transient in nature. It is normal to occasionally see such problems. Indeed, for connections to troublesome systems, it can even be common. So the occasional such error is usually nothing to be concerned about. However, if you are consistently seeing such errors on most messages on a channel, or seeing such errors on most messages to or from a particular remote system, then the errors can be an indication of an underlying network problem.

If you need more information about an os_smtp_* error, enable debugging on the channel in question and get a debug channel log file showing details of the attempted SMTP dialogue; see Section 2.3.4.85. In particular, the timing of exactly when a network problem occurred during the SMTP dialogue tends to be suggestive as to what sort of network or remote side issue might be involved. In some cases, you can also want to do network level debugging (*e.g.*, TCP/IP packet tracing) to see what was sent or received over the wire.

34.3.7 Error in qu_init: Usage Level Requires PMDF-MTA Service

If the error message "Usage level requires PMDF-MTA service" is reported by PMDF components, then a PMDF-ACCESS license is loaded on a system that has a PMDF-MTA configuration, that is, a configuration including channels not allowed on a PMDF-ACCESS system.

If this system is supposed to be a PMDF-ACCESS system, then the problem is that the PMDF configuration has non-ACCESS, disallowed components in it. Either manually modify the PMDF configuration file to remove non-ACCESS channels, or use the `pmdf configure access` utility to generate a new PMDF-ACCESS configuration.

If this system is supposed to be a PMDF-MTA system rather than a PMDF-ACCESS system, then the problem is that a PMDF-ACCESS license has been loaded instead of or in addition to a PMDF-MTA license. Note that the (lesser functionality) PMDF-ACCESS license will override the PMDF-MTA license; you must remove the PMDF-ACCESS license in order for the PMDF-MTA license to take effect.

34.4 Common Problems and Solutions

The following section lists some of the more common problems encountered during PMDF installation, configuration, and operations.

34.4.1 Changes to Configuration Files or PMDF Databases Do Not Take Effect

If changes to your configuration, mapping, conversion, security, option, or alias files or to PMDF databases do not seem to be taking effect, check to see if you have a compiled configuration or that you have exited and restarted your mail user agent (*e.g.*, Pine) session, and that you have restarted any PMDF components that need to be made aware of the change. See Section 8.2 for additional discussion.

34.4.2 PMDF Sends Outgoing Mail, but Does Not Receive Incoming Mail

Most PMDF channels depend upon a slave, or server, channel program to receive incoming messages. For some transports supported by PMDF, in particular TCP/IP and UUCP, you need to make sure that the transport activates the PMDF slave program rather than its standard server. Replacing the native sendmail SMTP server with the PMDF SMTP server is normally performed as a post-installation task when first installing PMDF; see the appropriate edition of the *PMDF Installation Guide* for instructions.

For the multithreaded SMTP server, the startup of the SMTP server is controlled via the PMDF Dispatcher. The PMDF Dispatcher controls the starting up of an SMTP server or servers, according to your Dispatcher configuration. If the Dispatcher is configured to use a MIN_PROCS value greater than or equal to one for the SMTP service, then there should always be at least one SMTP server process running (and potentially more, according to the MAX_PROCS value for the SMTP service). The `pmdf process` command can be used to check for the presence of SMTP server processes; see Section 34.2.4. Also, the Dispatcher statistics web page can be viewed to check just what connections SMTP processes are currently handling.

34.4.3 POP and IMAP Clients Time Out

The first thing to check is whether the clients are timing out when they try to connect, or when they try to send mail. If the trouble is when users try to connect to read messages, then you need to investigate the POP or IMAP server, according to which sort of client is having difficulties. But note that POP and IMAP clients send mail out using SMTP and the SMTP port, so if the trouble is when users try to connect to send messages, then you need to investigate the SMTP server.

The POP, IMAP, and SMTP servers are controlled by the PMDF Dispatcher. For the server in question, check the Dispatcher configuration for the maximum number of servers allowed for that service (MAX_PROCS), and the maximum number of connections each individual server can handle (MAX_CONNECTIONS). Then check how many such server processes are actually running. If you have more than MAX_PROCS*MAX_CONNECTIONS users trying to connect simultaneously, then you can want to increase the total number of servers allowed, or perhaps the number of connections each server can handle—though note that allowing too many connections per server tends to degrade the performance for each particular connection.

Additionally, general system resource and quota problems will of course impact the servers. See, for instance, Section 34.4.4.

34.4.4 Time Outs on Incoming SMTP Connections

Timeouts on incoming SMTP connections are most often related to system resources and the allocation thereof.

Check how many simultaneous incoming SMTP connections you allow. This is controlled by the MAX_PROCS and MAX_CONNECTIONS Dispatcher settings for the SMTP service; the number of simultaneous connections allowed is MAX_PROCS*MAX_CONNECTIONS. If you can afford the system resources, consider raising this number if it is too low for your usage.

Try putting the `slave_debug` keyword on the channels handling incoming SMTP over TCP/IP mail, usually tcp_local. Then take a look at the resulting `tcp_local_slave.log-`*`uniqueid`* files, and try to spot any particular characteristics of the messages that time out. For instance, if incoming messages with large numbers of recipients are timing out, consider using the `expandlimit` keyword on the channel.

Of course, if your system is extremely overloaded and overextended, time outs will be difficult to avoid entirely.

34.4.5 Outgoing TCP/IP Messages Sit in Queue

Errors encountered during TCP/IP delivery are quite often transient in nature and PMDF will generally retain messages when problems are encountered and retry them periodically. It is quite normal on very large networks to experience periodic outages to certain hosts while other host connections work fine. You can examine the log files for errors relating to delivery attempts. You can see error messages such as "Fatal error from smtp_open". Such errors are not uncommon and are usually associated with a transient network problem. Your TCP/IP package can contain tools such as ping, traceroute, and nslookup to aid in debugging TCP/IP network problems.

Example 34–1 shows the steps you might use to see why a message is sitting in the queue awaiting delivery to xtel.co.uk. The basic idea is to duplicate the steps PMDF uses to deliver SMTP mail on TCP/IP.

Example 34–1 Tracing TCP/IP Mail Delivery

```
% nslookup -query=mx xtel.co.uk ❶

Server:  LOCALHOST
Address:  127.0.0.1

Non-authoritative answer:
XTEL.CO.UK      preference = 10, mail exchanger = nsfnet-relay.ac.uk ❷

% /usr/sbin/ping nsfnet-relay.ac.uk ❸
PING NSFNET-RELAY.AC.UK (128.86.8.6): 56 data bytes
64 bytes from 128.86.8.6: icmp_seq=0 time=490 ms
CANCEL

% telnet nsfnet-relay.ac.uk 25 ❹
Trying... [128.86.8.6]
telnet: Unable to connect to remote host: Connection refused
```

❶ First use the NSLOOKUP utility to see what MX records, if any, exist for this host. If no MX records exist, then you should try connecting directly to the host. If MX records do exist, then you must test by connecting to the designated MX relays since PMDF is required to honor MX information preferentially.

❷ In this example, the Domain Name Service returned the name of the designated MX relay for xtel.co.uk. This is the host that PMDF will actually connect to. If more than one MX relay is listed, PMDF would try each in succession.

❸ A simple way to test connectivity to the host is with a PING utility. If no response is received then you have a network routing or configuration problem. If the problem is on some router over which you have no control, there is not anything you can do except to wait until it is fixed.

❹ If you do have connectivity to the remote host, the next step is to see if it is accepting inbound SMTP connections by using TELNET to the SMTP server port, port 25. If you use TELNET without specifying the port, you can merely discover that the remote host accepts normal TELNET connections. This by no means indicates that it accepts SMTP connections: many systems can accept regular TELNET connections

but refuse SMTP connections or vice versa. Thus, you should always do your testing against the SMTP port.

In this example, the remote host is currently refusing connections to the SMTP port. This is undoubtedly why PMDF fails to deliver the message. The connection can be refused due to a misconfiguration of the remote host or some sort of resource exhaustion, again, on the remote host. There is absolutely nothing you can do locally to solve the problem. Typically, you should just let PMDF continue to retry the message.

If you are running on a TCP/IP network which does not use the Domain Name Service, then you can skip steps ❶ and ❷ and use PING and TELNET directly to the host in question. Be careful to use precisely the host name that PMDF would use, which can be ascertained by examination of the relevant log file from PMDF's last attempt.

Note that if you test connectivity to a TCP/IP host and encounter no problems using interactive tests, it is quite likely that the problem has simply been resolved since PMDF last tried delivering the message. This is not an indication of a problem with PMDF.

34.4.6 PMDF Messages are Not Delivered

In addition to message transport problems, there are two other common problems which can lead to messages sitting around unprocessed in the message queues:

1. There is a problem with the job controller process. It has crashed or is hung, or is having some other problem which prevents it from processing messages.
2. The job controller's in-memory queue cache database is not synchronized with the messages in the queue directories.

   Message files in the PMDF queue subdirectories which are awaiting delivery are entered into the job controller's in-memory queue cache database. When channel programs run in order to deliver messages in their queues they consult the job controller to determine what messages to process. There are circumstances which can lead to message files in the queue that are not known to the job controller. Channel programs will ignore queued messages which are not entered in the job controller's queue cache. You can use the `pmdf cache -view` utility to check if a particular file is in the queue cache; if it is not, then the queue cache needs to be synchronized.

   The queue cache is normally resynchronized daily. If required, you can manually resynchronize the cache using the UNIX command

   ```
   # pmdf cache -synchronize
   ```

   Once resynchronized, upon the next running of the PMDF periodic delivery job the channel programs should process all messages in their queue.

3. Channel processing programs fail to run because they cannot create their execution log file.

   Check the access permissions, disk space and quotas.

34.4.7 Message Queues Contain .HELD Files

Mail messages in the PMDF message queue directories generally have a two-digit file extension. If PMDF detects a message that is looping, it will rename the file so that it has an extension of `.HELD`. Message files with such a file extension will not be processed by PMDF channel programs and therefore will not be delivered. This is a safety mechanism to prevent messages from looping indefinitely. Looping messages are detected by having a large number of Received: headers lines.

34.4.7.1 Diagnosing .HELD Files

One cause of message loops is user error: a user forwards their messages on system A to system B, and has system B set up to forward back to system A. The solution is for the user to fix their forwarding definitions.

Another common cause of message loops is PMDF receiving a message that was addressed to your host with a network name that PMDF does not recognize as one of the host's own names. For example, imagine a host which is known to the TCP/IP domain name system and to other hosts and users as example.com, but whose PMDF configuration does not know that. A message is sent to joe@example.com and is accepted by the network and delivered to this host. Since PMDF does not know itself as example.com, it will likely assume that example.com is elsewhere and direct the message back out to the network and unwittingly loop the message back to itself. This loop will continue until PMDF detects the loop and puts the message on hold.

If you detect such a situation you should try to determine by examination of the message file whether there is a name you should add to your PMDF configuration as a synonym for your official local host name. The Received: lines should show the path the message travels through the loop.

Example 34–2 `pmdf.cnf` **For milan.example.com**

```
! pmdf.cnf - PMDF configuration file for milan.example.com
! Written by SYSTEM, 19-AUG-2012 21:23
! This file was created by the PMDF configuration generator.
!
! Rewrite rules for the local host/cluster
!
milan                                          $u@milan.example.com ❶
milan.example.com                              $u@milan.example.com
naples                                         $u@milan.example.com ❷
naples.example.com                             $u@milan.example.com
example.com                                    $u@milan.example.com ❸
!
! Rewrite rules for the Internet
                .
                .
                .
```

Example 34–2 Cont'd on next page

Example 34–2 (Cont.) `pmdf.cnf` **For milan.example.com**

```
l nox_env_to
milan.example.com ❹

                .
                .
                .
```

❶ Rewrite rules for first cluster member to official local host.

❷ Rewrite rules for second cluster member to official local host.

❸ We have added this rule so example.com is recognized.

❹ The host name on the L channel (local channel) is always the official local host.

In Example 34–2 we have added example.com as another name for the cluster consisting of milan.example.com and naples.example.com, where the official local host name has been milan.example.com. Mail addressed to joe@example.com will now be properly recognized and locally delivered.

If you do not believe that the name in question should be directed to your host, then you can have to address the problem with a network configuration change or by changing the behavior of a remote mailer.

34.4.7.2 Cleaning Up .HELD Files

After diagnosing and fixing the cause of the loop, `.HELD` files should be renamed to `.00`; for example, in the csh shell you can use commands such as the following:

```
# cd /pmdf/queue/tcp_local
# foreach N ({.,*}/*.HELD)
? mv $N `dirname $N`/`basename $N \.HELD`.00
? end
```

Then synchronize the queue cache with the command

```
# pmdf cache -synchronize
```

(Alternatively, the `pmdf qm -maint` utility's `release` command can be used to cause message files to cease to be `.HELD` and the PMDF queue cache database to be synchronized.)

Then use the `pmdf startup post` to run the PMDF periodic delivery retry job immediately. Now, after the resulting jobs have run, look around to see if there are new `.HELD` files. There can very well be some. If there are still `.HELD` files in the original queue directory, then you can not have solved the looping problem. However, you can also find `.HELD` files in other queue directories such as the local channel (L channel) queue directory. This is because PMDF marks a message `.HELD` when it has too many Received: lines in which the local host appears. As the message moved from the original directory to another directory (*i.e.,* moved from one channel to another), PMDF again saw too many Received: lines in the message's header and again marked it `.HELD`.

This is to be expected. Simply repeat the process of renaming the `.HELD` files to `.00`, synchronizing the queue cache, and again submitting PMDF processing jobs. Repeat this process until there are no more `.HELD` files in any of the channel queue directories.

34.4.8 Messages are Looping

If PMDF detects that a message is looping, that message will be sidelined as a `.HELD` file; see Section 33.4.7 for a discussion. But certain cases can lead to message loops which PMDF can not detect. Some of the more common cases include:

1. A postmaster address is broken.
2. Stripping of Received: headers is preventing PMDF from detecting the message loop.
3. Incorrect handling of notification messages by other mail systems, that are generating reencapsulated messages in response to notification messages.

The first step in dealing with looping messages is to determine why the messages are looping. Useful things to look at are a copy of the problem message file while it is in the PMDF queue area, PMDF mail log entries (if you have the `logging` channel keyword enabled in your PMDF configuration file for the channels in question) relating to the problem message, and PMDF channel debug log files for the channels in question. Determining the From: and To: addresses for the problem message, seeing the Received: headers, and seeing the message structure (type of encapsulation of the message contents), can all help pinpoint which sort of message loop case you are encountering.

For case (1), note that mail systems such as PMDF require that the postmaster address be a functioning address that can receive e-mail. If a message to the postmaster is looping, check that your configuration has a proper postmaster address pointing to an account that can receive messages.

For case (2), note that normal detection of message loops is based on various Received: headers. If Received: headers are being stripped—either explicitly on the PMDF system itself, or more likely on some other system such as a firewall—that interferes with proper detection of message loops. There will likely be two issues to resolve: check that no undesired stripping of Received: headers is occurring so that if a loop does occur it can be short-circuited, and check for the underlying reason why the messages were looping. Possible underlying reasons for the occurrence of the message loop in the first place include: a problem in the assignment of system names or a system not configured to recognize a variant of its own name, a DNS problem, a lack of authoritative addressing information on the system(s) in question, or a user address forwarding error.

For case (3), note that Internet standards require that notification messages (reports of messages being delivered, or messages bouncing) have an empty envelope From: address to prevent message loops. However, some mail systems do not correctly handle such notification messages; such mail systems can, when forwarding or bouncing such a notification message, insert a new envelope From: address of their own. This can then lead to message loops. The solution is to fix the mail system that is incorrectly handling the notification messages.

34.4.9 Received Message is Encoded

Messages sent by PMDF are received in an encoded format; *e.g.,*

```
Date: Sun, 07 Jul 2012 11:59:56 -0700 (PDT)
From: "Elvis Presley" <elvis@example.com>
To: priscilla@example.edu
Subject: test message with 8bit data
MIME-Version: 1.0
Content-type: TEXT/PLAIN; CHARSET=ISO-8859-1
Content-transfer-encoding: QUOTED-PRINTABLE

2=00So are the Bo=F6tes Void and the Coal Sack the same?=
```

Such messages appear unencoded when read with a MIME-aware user agent such as Pine or when decoded with a decoder such as `pmdf decode`.

The SMTP protocol as set forth by RFC 821 only allows the transmission of ASCII characters. As ASCII is a seven-bit character set, the transmission via SMTP of eight bit characters is illegal. As a practical matter, the transmission of eight bit characters over SMTP is known to cause a variety of problems with some SMTP servers (*e.g.,* cause SMTP servers to go into compute bound loops, cause mail messages to be sent over and over again, crash SMTP servers, wreak havoc with user agents or mailboxes which cannot handle eight bit data, *etc.*).

Until the advent of RFC 1425 and RFC 1426, an SMTP client had only three alternatives when presented with a message containing eight bit data: return the message to the sender as undeliverable, encode the message, or send it anyhow in direct violation of RFC 821. None of these alternatives were pleasant; prior to version 4.2, PMDF chose the latter of the three owing to the lack of a standardized encoding format. However, with the recent advent of MIME (first specified in RFCs 1521 and 1522, and updated in RFCs 2045–2049) and the SMTP extensions work (RFC 1425 and RFC 1426), there are now standard encodings which can be used to encode eight bit data using the ASCII character set and mechanisms to negotiate, between the SMTP client and server, whether or not eight bit data will be accepted as is by the server without first being encoded.

When recipients receive encoded messages such as those shown above with a MIME content type of TEXT/PLAIN, then invariably the original message contained eight bit characters and the remote SMTP server to which the PMDF SMTP client transferred the message did not support the transfer of eight bit data. PMDF then had to encode the message.

34.4.10 From: Address Missing in Notifications from PMDF

Occassionally users or postmasters on other mail systems will complain that PMDF is losing, dropping, forgetting, or otherwise omitting the envelope From: address in messages it sends. You can be presented with a message header fragment like the one shown below

```
From   Thu Jul 11 11:50:23 2012
Received: from vulcan.ajax.com by monster.ajax.com via SMTP
  (930416.SGI/931108.SGI.ANONFTP) for xxxx id AA21154;
  Thu, 11 Jul 02 11:50:23 +1100
Date: Thu, 11 Jul 2012 11:49:26 +1000
From: PMDF Mail Server <postmaster@vulcan.ajax.com>
```

Note how in the first line there is a noticeable blank space between the "From" and date? This header line is often referred to as the "colonless From line" and it gives the envelope From: address for the message. That blank space indicates that the message had no envelope From: address; that is, it had what is called in the mail business a "null return path". Note further that this was an automatically generated mail message as suggested by the RFC 822 From: address of postmaster@vulcan.ajax.com.

The relevant standards require that automatically generated messages such as non-delivery notifications and delivery receipts use a null return path. As mailers are supposed to bounce mail to the envelope From: address,[3] this helps to prevent mail loops from occurring.

If someone complains about the missing From: address, ask them to send you a sample offending message. Determine if it was an automatically generated message. If it was, then explain to them that if their mailer or user agent is incapable of handling null return paths then it is incompliant with RFC 821 and 1123. Refer them to Paragraph 8 of Section 3.6 and the second paragraph of the MAIL command description in Section 4.1.1 in RFC 821. Further point out that were you to change your mailer to use a non-null return path for automatically generated notifications, then you would be violating the Internet Host Requirements; specifically, you would be in violation of Section 5.3.3 of RFC 1123.

Now, if for some reason you absolutely must generate non-null return paths in your notification messages, then you can do so with the RETURN_ENVELOPE option of the PMDF option file; see Section 7.3.4. Or to generate non-null return paths in notification messages only for a particular channel or channels, you can use the `returnenvelope` channel keyword; see Section 2.3.4.64. Be warned: Use of either the option or the channel keyword will put you in violation of the Internet Host Requirements and, more importantly, can lead to looping mail. Looping mail will not only inconvenience you but can cause serious problems for some unfortunate site which gets into a loop with your system. Also, keep in mind that changing PMDF's behavior so as not to cause problems for a broken mailer which cannot handle null return paths does not really fix anything: Other mailers over which you have no control will continue to send the broken mailer messages with null return paths. The only satisfactory solution in this situation is to fix the broken mailer.

[3] Some mailers will preferentially send notifications to the address specified with the non-standard Errors-to: or Warnings-to: header lines. By default, PMDF itself sends notifications to the envelope From: address, unless configured otherwise via the USE_ERRORS_TO and USE_WARNINGS_TO PMDF options.

34.5 Contacting Process Software Technical Support

Process Software provides technical support only for sites with a current maintenance agreement.

If you obtained PMDF from an authorized Process Software distributor, then technical support in your timezone can be most efficiently obtained from your distributor. You can also contact Process Software directly if you want.

Process Software technical support can be contacted at:

Process Software, LLC
959 Concord Street
Framingham, MA 01701 USA
+1 508 879 6994
+1 508 879 0042 (FAX)
support@process.com

The public mailing list for PMDF sites, info-pmdf@process.com, is another useful source of input and advice from other sites using PMDF.

When reporting a question or problem to Process Software technical support:

- Please include the output of the `pmdf version` command. This will include the architecture type for your system, the operating system version for your system, the PMDF installed version number, and the version of your PMDF shared library image.
- If the question regards a message, please include an *entire* sample message, in particular including all message headers. When sending a sample message via e-mail, please extract the sample message (with all headers) to a file and then send that entire file as an attachment; this is much preferable to forwarding the message as a message since forwarding as a message leaves headers (which can be an essential clue) subject to further processing.
- If the question regards the operation of a particular channel, please include a debug channel log file; see Section 2.3.4.85 for a discussion of obtaining channel debug log files.
- Be prepared to send copies of relevant PMDF configuration files, if Process Software technical support requests them.

Volume IV

The *PMDF System Manager's Guide* is in four volumes. Volume I comprises Chapter 1 through Chapter 13. Volume II comprises Chapter 14 through Chapter 28. Volume III comprises Chapter 29 through Chapter 34.

PMDF software products are marketed directly to end users in North America, and either directly or through distributors in other parts of the world depending upon the location of the end user. Contact Process Software for ordering information, to include referral to an authorized distributor where applicable:

Process Software, LLC
959 Concord Street
Framingham, MA 01701 USA
+1 508 879 6994
+1 508 879 0042 (FAX)
sales@process.com

Glossary

APOP: RFC 1939, POP3, defines the APOP command. This is an alternate method for authenticating the user which, rather than sending the username and password in the clear over the network, encodes the password.

Authentication mechanism: An authentication mechanism is a particular method for a client to prove its identity to a server. APOP, PLAIN and CRAM-MD5 (mechanism names are as defined by RFC 2222, SASL), are examples of authentication mechanisms. Another, but non-standard, mechanism is the LOGIN mechanism. For discussions of particular mechanisms, see for instance RFC 2195 documenting CRAM-MD5, and RFC 1939 documenting APOP.

Authentication source: An authentication source is a file, database, interface to an LDAP directory, *etc.*, accessible to the server wherein are stored authentication verifiers for users. The system password file, PMDF user profile passwords (PMDF MessageStore or PMDF popstore profile passwords) and the PMDF password database are examples of authentication sources.

Authentication verifier: An authentication verifier (*e.g.*, password) is stored on the server and contains information used to verify a user's identity. The format of the authentication verifier can restrict which mechanisms can be used. The term authentication verifier is preferred in place of password, since while passwords are the most common instance of authentication verifiers, an authentication verifier could also be something like a certificate in an LDAP directory or the like; usually, however, one can think "password" wherever one sees "authentication verifier".

Certificate: In the security context, a certificate is a guarantee, signed by some trusted authority, that says that a piece of information is what it purports to be. For instance, certificates are often needed and encountered when using a *public key pair*.

Certificate Authority: A Certificate Authority is a recognized, generally well-trusted authority that is willing to *sign* other organization's certificates. A Certificate Authority has a well-published *certificate* (containing their *public key*) that other organizations can use to verify the Certificate Authority's signature on other certificates. Assuming that an organization trusts the Certificate Authority, this then gives them confidence in certificates that include a valid signature from the Certificate Authority. Verisign, Inc., and Thawte Consulting are two of the better known commercial Certificate Authorities. Large corporations will sometimes, for their own internal purposes, act as their own Certificate Authority.

Certificate request: A certificate request is a special form of a site's *public key* suitable for signing by a Certificate Authority. The signing of a certificate request generates a *certificate*.

Channel block: The definition of a PMDF channel appearing in the PMDF configuration file is called a channel block; see Section 2.3.2.

Channel keyword: A large number of channel keywords are available for use in PMDF channel definitions (channel blocks) to control and modify the action of the channel to which a keyword is applied; see Section 2.3.4.

Channel program: Loosely speaking, any program which enqueues or dequeues messages to or from PMDF's message queues.

Common name: In X.500 terminology, the common name or commonName or CN attribute is a multi-valued attribute that describes an entry; typically it is something like a person's name, "First Middle Last", *etc*. The term is also used in other contexts, such as in a *certificate*, and in non-X.500 directories, especially LDAP directories.

CRAM-MD5: RFC 2195, IMAP/POP AUTHorize Extension, defines the CRAM-MD5 mechanism (Challenge-Response Authentication Mechanism using the MD5 digest algorithm) for authenticating using an encoded password, rather than sending the user's password in the clear over the network.

Dequeue: The act of removing a mail message from PMDF's message queues.

Distinguished name: In X.500 terminology, the distinguished name or distinguished-Name or DN attribute uniquely identifies an object in the Directory Information Tree. The term is also used in other contexts, such as in a *certificate*, and in non-X.500 directories, especially LDAP directories, and hence is a much used term in PMDF-DIRSYNC.

Enqueue: The act of submitting for transmission a mail message to PMDF.

Envelope: The message's transport layer To: and From: addressing information is contained in the message envelope.

GUI: A Graphics User Interface or GUI is a visually oriented interface, such as typically seen on Mac or Windows systems.

IETF: The IETF, Internet Engineering Task Force, is the Internet standards body.

Keyword: See Channel keyword.

Private key: A private key is the secret half of a *public key pair*.

Public key: A public key is the published half of a *public key pair*.

Public key encryption: So-called public key encryption refers to encryption and decryption using a pair of keys, referred to as a *public key pair*. One key is referred to as the *public key*, and is generally published (visible to the outside world); the other key is referred to as the *private key* and is secret and known only to the owner of the public key pair. User A can encrypt data to send to user B using user B's public key, and then only user B will be able to decrypt the data by using user B's own private key.

Public key pair: Public key encryption uses pairs of keys, one kept secret and one published (made accessible) to the outside world. What the public key encrypts, the private key decrypts, and *vice-versa*. The keys together are called a public key pair.

Mailbox filter: Mailbox filters are rules for individual users, for PMDF channels, or for the PMDF system specifying screening of incoming messages; see Section 16.2.

Mapping table: Many components of PMDF make use of one or another mapping table: a table mapping input strings to output strings. All PMDF mapping tables are stored in the PMDF mapping file; see Chapter 5.

Master channel program: Any program which enqueues messages to PMDF's message queues.

MIME: See RFCs 2045–2049.

MTA: Message transfer agent; *e.g.,* PMDF.

MUA: Mail user agent; see UA.

NOTARY: See RFCs 1891–1894.

RFC: Request For Comments; the Internet's method of publishing documents.

RFC 821: RFC 821, written by Jonathan Postel, defines SMTP, the Simple Mail Transfer Protocol, used to transfer messages over the Internet.

RFC 822: RFC 822, written by David Crocker, is the Internet standards document entitled *Standard for the Format of ARPA Internet Text Messages*. Messages in PMDF's message queues conform to this standard; *i.e.,* RFC 822 is the format which PMDF uses internally.

RFC 1123: RFC 1123, edited by Robert Braden, is the Internet standards document entitled *Internet Host Requirements — Application and Support*. PMDF adheres to the requirements put forth by this document.

RFC 1566: RFC 1566, sometimes referred to as MADMAN, written by Steve Kille and Ned Freed, is the Internet standards track protocol entitled *Mail Monitoring MIB*. PMDF accumulates the necessary message traffic statistics needed for this MIB. The concept of "group" used in the MIB is identified with a PMDF channel. The PMDF_get_channel_stats routine can be used to access the messages traffic statistics, referred to as channel statistics.

RFCs 1891–1894: RFCs 1891–1894, sometimes referred to as NOTARY, written by Keith Moore and Greg Vaudreuil, are the Internet standards track documents for the format and handling of notification messages.

RFCs 2045–2049: RFCs 2045–2049, commonly referred to as MIME, written by Nathaniel Borenstein and Ned Freed, are the Internet standards track documents describing the format of Internet message bodies. PMDF uses the specifications laid out in this document when forming multipart messages, encoded messages, *etc.* Note that RFCs 2045–2049 replaced RFCs 1521–1522 and 1431, previous drafts of MIME.

RFC 2222: RFC 2222, SASL (Simple Authentication and Security Layer), describes methods for adding authentication mechanisms, encryption, and data integrity checking to protocols such as POP, IMAP, and SMTP.

RFC 2246: RFC 2246 defined TLS (Transport Layer Security), a protocol for providing data integrity and encryption for reliable data connections (such as TCP).

SASL: See RFC 2222.

Security rule set: In the PMDF security configuration file context, a security rule set is a set of rules determining which authentication mechanisms and sources are permitted or used by the server. In PMDF the PORT_ACCESS mapping is used to determine the security rule set to apply to an incoming connection, based on IP addresses and ports.

Sign: In the context of a TLS certificate, to say that a certificate is signed means that a Certificate Authority has generated a hash of the contents of the certificate and used their own private key to encrypt that hash and then appended that encrypted hash to the original certificate. Then other sites that want to check on the validity of your certificate can use the Certificate Authority's well-published certificate (containing the Certificate Authority's public key) to verify the hash and thus verify that the contents of your supposed certificate match the contents that were seen and signed off on by the Certificate Authority. Assuming that the other site trusts the Certificate Authority, this then gives them confidence in your certificate.

Slave channel program: Any program which dequeues messages from PMDF's message queues.

SMTP (Simple Mail Transfer Protocol): See RFC 821.

SSL (Secure Sockets Layer): This protocol was developed by Netscape and has been superseded by TLS which is backward compatible with SSL.

Symmetric encryption: When encryption is done such that the same key encrypts and decrypts the data, it is said that the encryption is symmetric. This does require that the key that is used to encrypt the data must be given to the decryption side in a secure fashion.

TLS (Transport Layer Security): See RFC 2246.

UA: User agent; *e.g.,* the VMS MAIL utility or the Pine utility.

User domain: A user domain is an independent set of users known to the server. This is useful, for example, if a server wants to support multiple sets of users possibly with overlapping user names. In PMDF the PORT_ACCESS mapping is used to determine the user domain for each incoming connection, based on IP addresses and ports. Currently only the PMDF popstore authentication source supports multiple user domains; for all other sources, or if no user domain is explicitly specified in the PORT_ACCESS mapping, the default user domain is assumed. Currently only the PMDF POP server supports authentication using a user domain.

Virtual domain: When a system hosts multiple domain names, it is considered to be supporting virtual domains—pseudodomain names that do not correspond to a system dedicated to only that domain name. PMDF's directory channel, *etc.*, and user domain support for PMDF popstore users, are examples of PMDF features helpful in supporting virtual domains.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X