HP MAILbus 400 Message Transfer Agent

Tuning and Problem Solving

October 2006

Revision/Update Information:	This is a revised manual.
Operating System and Version:	OpenVMS Alpha Version 7.3-2 and above OpenVMS VAX Version 7.3
Software Version:	HP Mailbus 400 Message Transfer Agent V3.2-12 for OpenVMS

Hewlett-Packard Company
Palo Alto, California

Printed in the US

The HP OpenVMS documentation set is available on CD-ROM.

This document was prepared using DECdocument, Version 3.3-1B.

Contents

Tables

Preface

Purpose of this Guide

This guide describes how a MAILbus 400 Message Transfer Agent works, how to optimize the performance of a MAILbus 400 MTA and how to solve problems that can occur in your Message Transfer System (MTS).

This guide provides the information that you require to support a MAILbus 400 MTA, within a mixed MTS of:

- Other vendors' messaging products
- Other MTAs

This guide does not describe how to solve problems in an MTS consisting solely of other vendors' products, or other HP products.

Structure of this Guide

This guide is divided into three parts:

- Part I, How the MTA Works

  This part describes how a MAILbus 400 MTA works and outlines the management of a MAILbus 400 MTA.
- Part II, Tuning

  This part describes how to modify the performance of a Message Transfer System made up of MAILbus 400 MTAs so that it fulfills your system or management requirements. This part also describes how to collect information about the message traffic in your routing domain and how to collect events.
- Part III, Solving Problems

  This part describes the problems that you might encounter in an MTS consisting of MAILbus 400 MTAs and how to solve them.

The following appendixes describe information that is specific to the operating system where the MTA is installed:

- Appendix D for Tru64 UNIX specific information.
- Appendix E for OpenVMS specific information.

Where appropriate, the text within this guide references this as "the appendix describing the operating system specific information".

Prerequisites

To manage and tune one or more MAILbus 400 MTAs and to solve problems that can occur in an MTS, you should have knowledge of DECnet/OSI®. You should also be familiar with HP's Network Control Language (NCL) and with the concepts of HP's Enterprise Management Architecture (EMA).

Related Documents

In addition to this guide, the documentation provided with the MAILbus 400 MTA comprises:

- *MAILbus 400 Getting Started*
- *HP MAILbus 400 MTA Planning and Setup*
- HP MAILbus 400 MTA installation documentation
- *HP MAILbus 400 MTA Software Product Description* (SPD) for the appropriate operating system
- *HP MAILbus 400 MTA Cover Letter*, where applicable, for the appropriate operating system
- Online Release Notes

Reference information is available in the *MTA Module Online Help* and *MTS Module Online Help*.

Release notes are also available online. For information about how to access the release notes, see the MAILbus 400 MTA installation documentation.

All the standards, recommendations, and profiles that are referenced in the MAILbus 400 MTA documentation set are listed in Appendix A.

You are advised to have available the following associated documentation:

- The HP Enterprise Directory Service documentation set and *Directory Module Online Help*.
- The HP DECnet-Plus documentation.

Conventions

The following conventions are used in this guide:

`this typeface`	Indicates prompts and messages from the computer and command examples.
`this typeface`	Indicates commands or responses that you type. Unless otherwise stated, press Return after each command or response.
`variable`	Represents a variable.
UPPERCASE and lowercase	The Tru64 UNIX operating systems differentiate between lowercase and uppercase characters. Literal strings that appear in text, examples, syntax descriptions, and function descriptions must be typed exactly as shown.
newterm	Indicates the introduction of a new term.
#	A number sign (#) is the default Tru64 UNIX superuser prompt.
$	A dollar sign ($) is the default OpenVMS prompt.
Tru64 UNIX	Indicates the start of information that is applicable only to the Tru64 UNIX operating systems.
OpenVMS	Indicates the start of information that is applicable to the OpenVMS Alpha and OpenVMS VAX operating systems.
OpenVMS VAX	Indicates the start of information that is applicable only to the OpenVMS VAX operating systems.

OpenVMS Alpha	Indicates the start of information that is applicable only to the OpenVMS Alpha operating systems.
♦	Indicates the end of information that is applicable to a particular operating system.
(Tru64 UNIX)	Where text that refers exclusively to the Tru64 UNIX operating system is minimal, the operating system is indicated in brackets after the text.
(OpenVMS)	Where text that refers exclusively to the OpenVMS Alpha or OpenVMS VAX operating systems is minimal, the operating system is indicated in brackets after the text.

Examples Used in This Guide

Command Examples:

In this guide, all command examples are Network Control Language (NCL) commands unless otherwise stated.

The component `NODE "`*`node-id`*`"` within Network NCL examples refers to the node where the MTA that you want to manage is located.

If the MTA that you want to manage is located on your node, you can omit `NODE "`*`node-id`*`"` from NCL commands, or use `NODE 0`. See the DECnet/OSI documentation for more information about NCL command syntax.

Examples of Events and Entities:

The layout of events and entities on a screen varies according to the operating system that the MTA is running on. Therefore, the examples given in this guide may differ from the way events and entities are displayed on your system.

Abbreviations

The following abbreviations are used in this guide:

ACSE	Association Control Service Element
ADMD	Administration Management Domain
ANSI	American National Standards Institute
API	Application Program Interface
ASCII	American Standard Code for Information Interchange
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CCITT[1]	International Telegraph and Telephone Consultative Committee
CEN	Comité Européen de Normalisation
CENELEC	Comité Européen de Normalisation Electrotechnique
CLNS	Connectionless Network Service
CONS	Connection Oriented Network Service
CDA	Compound Document Architecture
DAP	Document Application Profile
DDA	Domain Defined Attribute
DDIF	Digital Document Interchange Format
DIS	Draft International Standard
DISP	Draft International Standardized Profile
DL	Distribution List
DOTS	Data Object Transport Syntax
DSA	Directory System Agent
DTE	Data Terminal Equipment
DTIF	Digital Tabular Interchange Format
DUA	Directory User Agent
EDI	Electronic Data Interchange
EIT	Encoded Information Type
ENV	Europa Norm Vorläufig (European Prestandard)
ETSI	European Telecommunications Standards Institute

[1]The CCITT is now the ITU–T (International Telephone Union—Telecommunications). Their published documents still have CCITT identification material, and to avoid confusion this book still uses the term CCITT.

EWOS	European Workshop for Open Systems
GDI	Global Domain Identifier
GOSIP	Government OSI Profiles
IEC	International Electrotechnical Commission
INTAP	Interoperability Technology Association for Information Processing Japan
IPM	Interpersonal Message
IPMS	Interpersonal Messaging System
IRV	International Reference Version
ISO	International Organization for Standardization
ISP	International Standardized Profile
ISPICS	International Standardized Profile Implementation Conformance Statement
LAN	Local Area Network
MHS	Message Handling System
MOTIS	Message Oriented Text Interchange System
MPDU	Message Protocol Data Unit
MTA	Message Transfer Agent
MTS	Message Transfer System
NIST	National Institute for Standards and Technology
NSAP	Network Service Access Point
ODA	Office Document Architecture
ODIF	Office Document Interchange Format
OIW	OSE Implementors' Workshop
O/R	Originator/Recipient
OSE	Open Systems Environment
OSI	Open Systems Interconnection
PICS	Protocol Implementation Conformance Statement
PRMD	Private Management Domain
PTT	Post Telegraph and Telephone Authority
RFC	Request For Comments
RTSE	Reliable Transfer Service Element

SCID	Session Connection Identifier
TCP/IP	Transport Control Protocol/Internet Protocol
TSEL	Transport Service Selector
WAN	Wide Area Network

Reader's Comments

HP welcomes your comments on this manual. Please send comments to either of the following addresses:

Internet	**openvmsdoc@hp.com**
Mail	Hewlett-Packard Company OSSG Documentation Group, ZKO3-4/U08 110 Spit Brook Rd. Nashua, NH 03062-2698

However, any software problem that requires action on the part of HP must be reported to your HP support center, as described in Chapter 23. Do not report such problems to the above addresses or those listed on the template.

How To Order Additional Documentation

For information about how to order additional documentation and for online versi on of this documentation, visit the following World Wide Web address:

`http://www.hp.com/go/openvms/doc/`

Part I

How the MTA Works

This part describes how a MAILbus 400 MTA works. Its purpose is to provide background knowledge to help you manage and support the product.

Chapter 1 gives an overview of the functions of a MAILbus 400 MTA and the entities of the MTA module that are used to manage them.

Chapter 2 describes the different ways in which messages enter an MTA.

Chapter 3 describes the sequence of operations performed by the message processing functions of the MTA.

Chapter 4 describes the different ways in which messages leave an MTA.

Chapter 5 gives an overview of the record keeping functions of the MTA during message transfer.

Chapter 6 describes how an MTA downgrades a message encoded according to the 1992 MHS Standards so that it is compatible with 1984 MHS Standards.

1

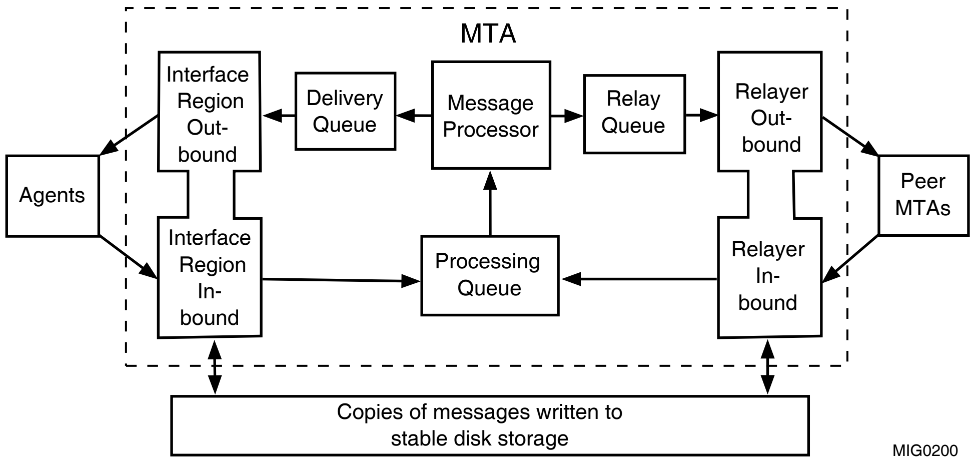
The MAILbus 400 MTA

This chapter introduces the main functions of the MAILbus 400 MTA. The functions are described in more detail in later chapters. This chapter also describes the hierarchy of the MTA module entities, and shows how you use attributes of these entities to manage MTA functions.

1.1 MTA Functional Design

The main functions of a MAILbus 400 MTA are represented in Figure 1–1.

Figure 1–1 Functional Overview of the MAILbus 400 MTA



Although, for the purposes of this guide, it is useful to represent the MTA in terms of its functions, it is important to note that you cannot manage the MTA directly in terms of these functions. You manage the MTA through the MTA module by setting attributes of the MTA entity and its subordinate entities, as described in Section 1.3.

1.1.1 The Interface Region

The **Interface Region** is the interface between Agent applications (User Agents, Gateways, and Message Stores) and the MTA. For inbound MPDUs, the Interface Region is responsible for submission and import. (An MPDU is an instance of a message, probe, or report in the MAILbus 400 MTA.) For outbound MPDUs, the Interface Region is responsible for delivery and export. HP supplies the **MAILbus 400 Application Program Interface** (API), which is a callable interface that enables users to write new User Agents or Gateways that can communicate with the MTA through the Interface Region.

An Agent written using the MAILbus 400 Application Program Interface can connect to the Interface Region of the MTA either directly or through the API server.

The **API Server** is provided with the MTA on the system where the MTA is running and it works on behalf of the Agent to issue calls to the MTA. This reduces the amount of memory and processing capacity required on the system where the Agent is located, as most of the processing is done on the system where the MTA is located.

Whether an Agent connects to the MTA directly or through the API Server is determined by how the Agent is linked against the MAILbus 400 API when it is built, and by where the Agent is running in relation to the MTA:

- If the Agent is linked against the *full* shareable image or library, it can connect directly to the MTA or through the API Server.
- If the Agent is linked against the *reduced* shareable image or library, it can only connect to the MTA through the API Server.
- If the Agent is running on a system remote from the MTA, it can only connect to the MTA through the API Server.

You can find out how Agents connect to your MTA by examining the /var/mta/mta_api_server_address file on Tru64 UNIX systems, or the value for the MTA_NODE logical on OpenVMS systems. Refer to Part III of *HP MAILbus 400 MTA Planning and Setup* for more information about setting up connections from Agents to the MAILbus 400 MTA.

Also provided with the MAILbus 400 MTA is the **Shared File** interface. When an Agent connects to the MTA using this interface, the Interface Region reads incoming messages, in the form of files, from an Input queue, and writes outgoing messages to files in an Output queue.

HP MAILbus 400 MTA Planning and Setup describes how to set up Agents for use with the MAILbus 400 MTA.

The Interface Region also deals with MPDUs that cannot be delivered immediately and those that cannot be delivered successfully within their defined expiry interval.

1.1.2 The Message Processor

The **Message Processor** is responsible for taking each incoming MPDU from the Processing Queue and preparing it for transfer to the next point in the MPDU's route to the eventual recipient. This involves obtaining routing data from the directory and, if necessary, converting and splitting the MPDU according to the information obtained. Once it has processed an MPDU, the Message Processor places it either on the Delivery Queue for use by the Interface Region or on the Relay Queue for use by the Relayer (see Section 1.1.3).

1.1.3 The Relayer

The **Relayer** is responsible for communicating with other MTAs. Other MTAs with which an MTA can communicate are known as **peer MTAs**. The Relayer can call a peer MTA to initiate an outbound association, or respond to a call from a peer MTA to establish an inbound association.

The MTA uses **associations** to transfer data, including MPDUs, to or from a peer MTA. An association is a connection between two applications in the Application layer, as defined by the **OSI Reference Model**. The OSI Reference Model is a conceptual model that describes an open systems architecture based on seven layers that handle the communications process.

The Relayer supports the Reliable Transfer Service Element (RTSE) of the OSI reference model. RTSE supports two types of dialogue mode – monologue and two-way alternate. The MAILbus 400 MTA supports only monologue mode. This means that it uses a single association either to send or to receive MPDUs, but not both at the same time.

Associations are created when necessary to transfer MPDUs between MTAs and can be used more than once. The Relayer can maintain associations in an idle state ready for further use, thus saving on the time and resources involved in establishing a new association. You can tune the MTA to modify the maximum period over which an association can remain idle. You can also modify other characteristic attributes of the MTA entity to optimize the flow of messages through it. Chapter 7 describes how to do this.

While transferring an MPDU, the Relayer keeps RTSE checkpoint information about the association in volatile storage. If, during transfer of an MPDU, an association fails and is later re-established, the Relayer can normally recover and complete the transfer by restarting it from the point where the checkpoint information shows it has been interrupted. Thus, checkpoint information is of use when an association fails due to network problems. However, because checkpoint information is held in volatile storage, it does not survive a failure or shutdown of the system supporting the MTA. In these circumstances, the Relayer cannot restart the transfer from the point at which it was interrupted. In the case of outbound transfer, the Relayer restarts transfer from the beginning of the MPDU. In the case of inbound transfer, it informs the other MTA that transfer must be restarted from the beginning of the MPDU.

If a peer MTA aborts an association recovery attempt, the Relayer discards the recovery information and restarts transfer from the beginning of the MPDU.

The Relayer also queues MPDUs that cannot immediately be transferred, and deals with MPDUs that cannot successfully be transferred within their defined expiry interval.

1.2 Queues

The MAILbus 400 MTA maintains a number of queues in memory, which hold MPDUs that are waiting to be processed, transferred, or delivered. These queues are:

- The Processing Queue, which holds MPDUs waiting to be processed by the Message Processor
- The Relay Queue, which holds MPDUs waiting to be transferred by the Relayer to another X.400 MTA
- The Delivery Queue, which holds MPDUs waiting to be delivered or exported through the Interface Region to the appropriate Agent

MPDUs are held in queues in order of their priority and arrival time. Queues are headed by MPDUs of the highest priority. For MPDUs of equal priority, those that arrive at the MTA first are at the front of the queue. The MPDU currently at the front of the queue is normally the first to be processed.

If the MAILbus 400 MTA fails to process an MPDU as intended, it returns the MPDU to the relevant queue for a further attempt.

Depending on the cause of the failure, the MAILbus 400 MTA tries to process the MPDU again until:

- Processing the MPDU has failed three times.

  If processing stops unexpectedly while the MPDU is being processed (for example, because of a problem within the MPDU), the MAILbus 400 MTA makes no further attempts to process the MPDU after failing three times.

- The MPDU expires.

  If the failure is due to a recognizable state, the MAILbus 400 MTA continues to attempt to process the MPDU until the MPDU reaches its expiry time (see Section 1.2.1).

When returning an MPDU to a queue, the MTA decides whether to assign a retry time to the MPDU to indicate the time at which the MTA can attempt to process the MPDU again. This depends on the queue to which an MPDU is returned and on the reason why the MPDU failed to be processed. Generally, the MTA assigns a retry time to all MPDUs returned to the Processing Queue and Relay Queue, while MPDUs are returned to the Delivery Queue without a retry time. However, the MTA can decide differently in certain circumstances where failure is due to a recognizable state.

Examples of recognizable states include the following:

- The directory is unavailable when the Message Processor attempts to look up address information. The MTA assigns a retry time to the MPDU and returns it to the Processing Queue.

- A peer MTA is unavailable when the Relayer attempts to transfer an MPDU. When the Relayer fails to transfer an MPDU to a peer MTA because the peer MTA is unavailable, a more complex retry mechanism is used. This is described in Section 4.3.2.

- During delivery an Agent requests that the MPDU be delivered at a later time. In this case the MTA assigns a retry time to the MPDU and returns it to the Delivery Queue.

If processing an MPDU fails unexpectedly three times, or if an MPDU in a queue expires, the MAILbus 400 MTA removes the MPDU from the queue and attempts to send a non-delivery report to the originator. In the former case, the MAILbus 400 MTA also copies the failed MPDU to a bad messages directory so that you can examine it using the Message Decoder utility (see Section 5.5).

1.2.1 Expiry Time

The expiry time of an MPDU depends on:

- The time set for the Local MPDU Expiry Interval attribute of the MTA.
- The CCITT domain expiry interval associated with the priority value of the MPDU. These expiry intervals are set by the following attributes of the MTA: Nonurgent MPDU Expiry Interval, Normal MPDU Expiry Interval, and Urgent MPDU Expiry Interval.

Priority-based MPDU expiry times are calculated from when the MPDU enters the first administration management domain (ADMD) on its route. Local expiry time is calculated from when the MPDU enters each MTA.

When an MTA has received an MPDU, the MTA periodically compares the expiry intervals with the time when the MPDU entered the MTA and the time when the MPDU entered the first ADMD. When one of the expiry intervals is reached, the MTA deletes the MPDU and sends a non-delivery report to the originator, if possible.

Guidelines for setting expiry intervals are given in Chapter 7.

1.3 The MTA Module

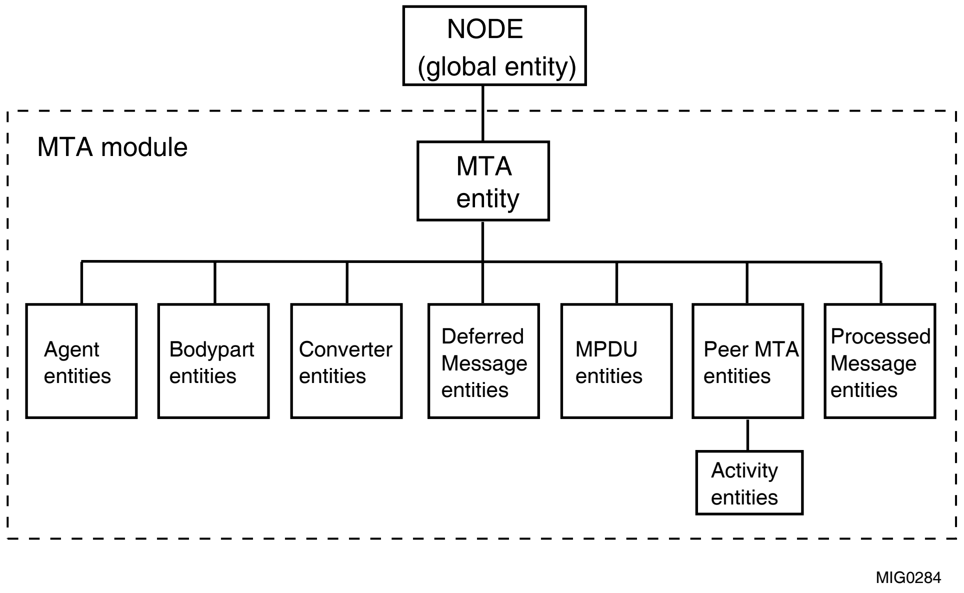
Within the Enterprise Management Architecture (EMA), you manage the MTA through the MTA module. There is one MTA module subordinate to the global entity NODE. The hierarchy of the MTA module is shown in Figure 1–2.

You manage the MTA by assigning values to characteristic attributes of the MTA entity and its subordinate entities. For example, you manage communication between the Interface Region of the MTA and Agent applications through attributes of the MTA entity and Agent entities. Similarly, you manage communication between the Relayer and peer MTAs in other routing domains through characteristic attributes of the MTA entity and Peer MTA entities.

You can obtain information about the status and performance of an MTA by showing the current values of status and counter attributes of the MTA entity and its subordinate entities.

EMA enables you to use Network Control Language (NCL) to manage an MTA from any node in a network. You need appropriate privileges or access rights to show or modify certain attributes. The *MTA Module Online Help* describes the specific commands used for MTA management.

Figure 1–2 The MTA Module in the EMA Entity Hierarchy

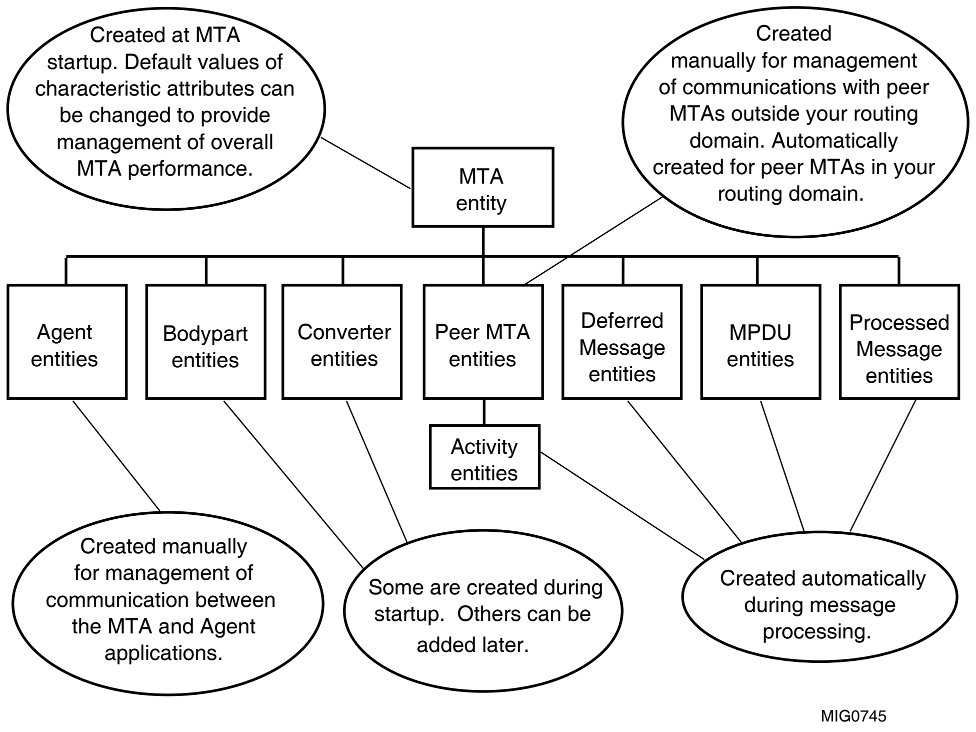


The remaining sections of this chapter describe the entities subordinate to the MTA entity. Figure 1–3 gives an overview of how and when the MTA entity and its subordinate entities are created.

1.3.1 Agent Entities

You use Agent entities to record information about registered Agent applications (User Agents, Gateways, or Message Stores) with which the MTA can connect. Registered Agents can support one or more O/R addresses and can be of two types. One type uses the XAPI interface, as implemented by the MAILbus 400 Application Program Interface; the other uses the Shared File interface (see *MAILbus 400 MTA Introduction and Glossary* or Section 1.1.1). Only Gateway Agents can use the Shared File interface.

Figure 1–3 How Entities are Created


Created at MTA startup. Default values of characteristic attributes can be changed to provide management of overall MTA performance.
MTA entity
Created manually for management of communications with peer MTAs outside your routing domain. Automatically created for peer MTAs in your routing domain.
Agent entities
Bodypart entities
Converter entities
Peer MTA entities
Deferred Message entities
MPDU entities
Processed Message entities
Activity entities
Created manually for management of communication between the MTA and Agent applications.
Some are created during startup. Others can be added later.
Created automatically during message processing.
MIG0745

Note

The MAILbus 400 MTA can also communicate with unregistered Agent applications that support only a single O/R address. These have no corresponding Agent entity and use the XAPI interface to identify themselves to the MTA using the O/R address they represent.

You have to create an Agent entity to record details (such as name and password) of a registered Agent. You can then use management directives to enable and disable the Agent entity in order to control the Agent's connection to the MTA.

1.3.2 Bodypart and Converter Entities

The contents of an interpersonal message (IPM) can consist of a number of IPM bodyparts. To make a bodypart acceptable to a recipient's Agent, it is sometimes necessary to convert it to a different format. The MTA has several converters that it uses to convert IPM bodyparts from one format to another. The MTA startup script that is supplied with the MAILbus 400 MTA creates corresponding Bodypart and Converter entities for these converters. You can create additional Bodypart and Converter entities for converters that you write. See the *MTA Module Online Help*, Chapter 11 and Chapter 12 for more details.

1.3.3 Peer MTA Entities

Peer MTA entities are created and configured either automatically or manually. They enable the management and monitoring of MPDU transfer between MTAs.

1.3.3.1 Automatically-Configured Peer MTA Entities

When two MTAs in the same routing domain first communicate with each other, each of them automatically creates a Peer MTA entity relating to the other peer MTA. These Peer MTA entities record information about communications between the two MTAs. Automatically-configured Peer MTA entities remain in existence until a limit on their total number is reached. Therefore, if two MTAs in the same routing domain have communicated in the past, the required Peer MTA entities might already exist.

The limit on the total number of Peer MTA entities is set by the Maximum Automatically Configured Peer MTAs attribute of the MTA (see Section 7.4). However, this is not a definitive limit. When the MTA needs to create a new Peer MTA entity, and the limit on the total number of Peer MTA entities is already reached, the MTA can delete existing automatically-configured Peer MTA entities. The MTA only deletes an automatically-configured Peer MTA entity if the Peer MTA entity does not have subordinate Activity entities (see Section 1.3.4 and Figure 1–3).

If all the MTA's automatically-configured Peer MTA entities have subordinate Activity entities, the MTA is unable to delete one. In this case, the MTA overrides the specified limit on the total number of automatically-configured Peer MTA entities.

If there are automatically-configured Peer MTA entities that can be deleted, the MTA deletes the Peer MTA entity that has been inactive for the longest time. A Peer MTA entity is considered inactive when it has no subordinate Activity entities.

1.3.3.2 Manually-Configured Peer MTA Entities

Boundary MTAs are MTAs in your routing domain that can communicate with peer MTAs in other routing domains. A boundary MTA can only communicate with a peer MTA in another routing domain if it has essential information about the peer MTA, such as its name, address, and password. To make this information available to the boundary MTA, you manually create and configure a Peer MTA entity with the required characteristic attributes at your boundary MTA. See *HP MAILbus 400 MTA Planning and Setup* for details of the information you need to plan in order to set up a Peer MTA entity, and the *MTA Module Online Help* for information about creating a Peer MTA entity.

Once you have created and configured a Peer MTA entity, you must enable it before the boundary MTA can communicate with the peer MTA in the other routing domain. You can subsequently disable and enable the Peer MTA entity to control the connection status of your MTA with the peer MTA in the other routing domain. When a manually-configured Peer MTA entity is enabled, it records information about communications with the peer MTA in the other routing domain.

You can modify a manually-configured Peer MTA entity if some information, such as the peer MTA password, changes. You can delete a manually-configured Peer MTA entity when it is inactive and communication with the relevant peer MTA is no longer needed.

1.3.4 Activity Entities

Activity entities are created automatically during MPDU transfer between peer MTAs. These entities provide information about the current state of the association and about a single MPDU being transferred. Although the MTA can use a single association with a peer MTA to transfer a number of MPDUs in succession, each Activity entity applies only to a single MPDU transfer. When the transfer is complete, the state of the corresponding Activity entity changes from active to idle. An idle Activity entity is deleted, and a new Activity entity is created, when another MPDU is transferred over the association.

If an association is broken during transfer of an MPDU, the state of the current Activity entity changes from active to interrupted, and the Activity entity remains in existence, to provide information for management and recovery purposes, until the association is re-established and the MPDU is transferred over it.

1.3.5 Deferred Message Entities

When a deferred message is submitted to the MTA, a Deferred Message entity is created automatically. Each Deferred Message entity represents an MPDU that has been marked by its originator for deferred delivery (see Section 2.2.1). The MTA holds such an MPDU, without processing it, until the deferred delivery time specified by the originator of the message.

1.3.6 MPDU Entities

An MPDU entity is created automatically when an MPDU enters the MTA. When processing an MPDU, the MTA may have to split the MPDU in order to create an individual occurrence of the MPDU for each conversion, O/R address or set of O/R addresses that represent a single destination (see Section 3.10). For each additional MPDU that the MTA creates in this way, the MTA also creates an MPDU entity.

1.3.7 Processed Message Entities

The MTA automatically creates Processed Message entities to record information about all messages (not probes or reports) from the time they enter the MTA. The MTA updates the attributes of Processed Message entities to record the current status of all MPDUs derived from an incoming message as they are processed and pass through the MTA.

If Message History logging is enabled, the information about the MPDUs that the MTA processes remains in stable disk storage until automatically purged. You can use the Message History Purge Interval attribute of the MTA entity to set the frequency with which the MTA purges Message History information.

If Message History logging is not enabled, the MTA does not keep records of MPDUs after it has finished dealing with them.

1.3.8 Creating Entities During MTA Startup

An MTA startup script is supplied with the MAILbus 400 MTA. You can customize the startup script so that it creates additional Agent, Peer MTA, Bodypart, and Converter entities specific to your particular requirements.

For the location of the MTA's startup script, refer to the appendix describing the operating system specific information. See the *MTA Module Online Help* and Part III of *HP MAILbus 400 MTA Planning and Setup* for information about how to edit this file.

The MTA startup script is not automatically updated with changes you make through management of the MTA. If, during interactive management of the MTA, you modify any entity attributes, you should reflect all these changes in the startup script if you want them to remain in effect next time the MTA is started up. If you do not update the MTA startup script, in the event of a system failure, you will not be able to restore the MTA automatically to the state it was in before the failure. Any MTA created will have only those settings contained in the latest copy of the startup script. It is therefore important to keep a secure copy of your latest customized version of the MTA startup script.

2
How MPDUs Enter the MTA

This chapter describes the different ways in which MPDUs enter the MTA and how the MTA takes responsibility for them.

2.1 Types of Entry into the MTA

A message, probe, or report enters the MTA by one of the following routes:

- Submission

  This is where an MPDU, representing a message or a probe, enters the MTA through the Interface Region from a User Agent or Message Store (see Section 2.2). Because the MPDU is entering the message transfer system (MTS) for the first time, its envelope carries no trace information relating to previous routing stages.

- Import

  This is where an MPDU, representing a message, probe, or report, enters the MTA through the Interface Region from a Gateway (see Section 2.2).

  The envelope of an imported MPDU can contain trace information that identifies previous stages in its transfer through the message handling system (MHS).

- Inbound transfer

  This is where an MPDU, representing a message, probe, or report, enters the MTA through the Relayer from another MTA (see Section 2.3).

2.2 MPDUs Entering Through the Interface Region

When an Agent has a message for transfer, it initiates an inbound communication to its MTA in the form of a submission or an import. When handling a submission or import, the Interface Region does the following:

1. Creates an MPDU with a local identifier and arrival timestamp.
2. Writes the MPDU to stable disk storage so that it can be recovered, if necessary (see Section 2.4 and Section 5.1).
3. Accepts responsibility for the MPDU.
4. Creates a Deferred Message entity, if an MPDU submitted from a User Agent has a deferred delivery time.

   If there is a deferred delivery time specified in the envelope of the MPDU, the Interface Region holds the MPDU until the specified time before adding it to the MTA's Processing Queue. See Section 2.2.1 for further information about MPDUs with deferred delivery times.
5. Adds the MPDU to the MTA's Processing Queue.

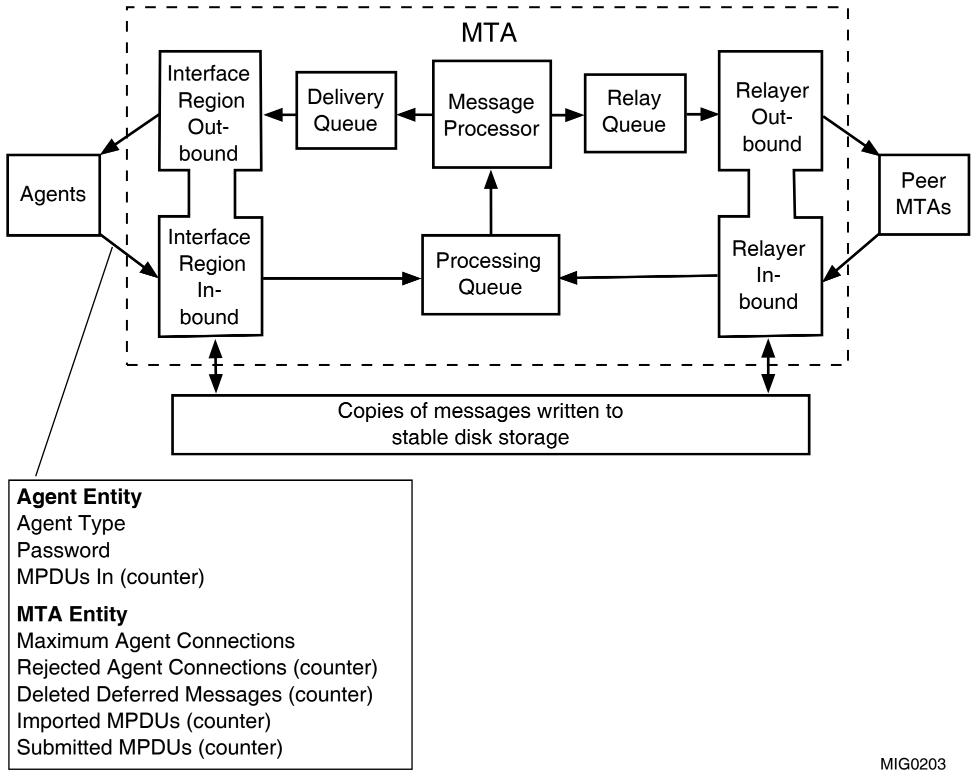
   This occurs as soon as the deferred delivery time has been reached, or immediately if no deferred delivery time has been specified.

Inbound communications with Agents are subject to conditions set by the values of some characteristic attributes of the MTA entity and Agent entities as shown in Figure 2–1. These may be values supplied with the product, or values you set when tuning the MTA. You can set a limit on the total permitted number of concurrent connections with Agent applications by setting the Maximum Agent Connections attribute of the MTA entity.

Figure 2–1 also shows counter attributes that you can show to monitor inbound communications with Agent applications.

If there are problems with a connection from an Agent, or with an MPDU already received, the MTA generates appropriate events (see Chapter 18 and Chapter 19).

Figure 2–1 Management of the Interface Region - Inbound



Agent Entity
Agent Type
Password
MPDUs In (counter)

MTA Entity
Maximum Agent Connections
Rejected Agent Connections (counter)
Deleted Deferred Messages (counter)
Imported MPDUs (counter)
Submitted MPDUs (counter)

2.2.1 Deferred Delivery

Deferred delivery allows the originator of a message to specify a future date and time at which MPDUs should be processed for transfer to the recipients.

If, during submission, the Interface Region detects a deferred delivery value in the MPDU envelope, it automatically creates a Deferred Message entity. It then holds the message to await processing at the time specified for delivery. When this time is reached, the Interface Region passes the message to the Processing Queue, as described in Chapter 3.

You can display the attributes of Deferred Message entities, but you cannot change them. You can delete a Deferred Message entity up to the time at which the message would be processed.

2.3 MPDUs Entering Through the Relayer

When the Relayer receives an inbound transfer association, it does the following:

1. Creates an MPDU with a local identifier and creation timestamp.
2. Writes the MPDU to stable disk storage so that it can be recovered if necessary (see Section 2.4 and Section 5.1).
3. Adds the MPDU to the Processing Queue.
4. Accepts responsibility for the MPDU.

An inbound association is initiated by a peer MTA that has an MPDU for transfer to the MTA. Inbound associations with peer MTAs are subject to conditions set by the values of the characteristic attributes of the MTA entity shown in Figure 2–2. These may be values that are supplied with the MTA or those that you set when tuning the MTA. For example, you can set a limit on the total permitted number of concurrent associations with peer MTAs by setting the Maximum Transfer Associations attribute of the MTA entity. In the case of boundary MTAs, associations with MTAs in other routing domains are also subject to conditions set by some characteristic attributes of manually configured Peer MTA entities.

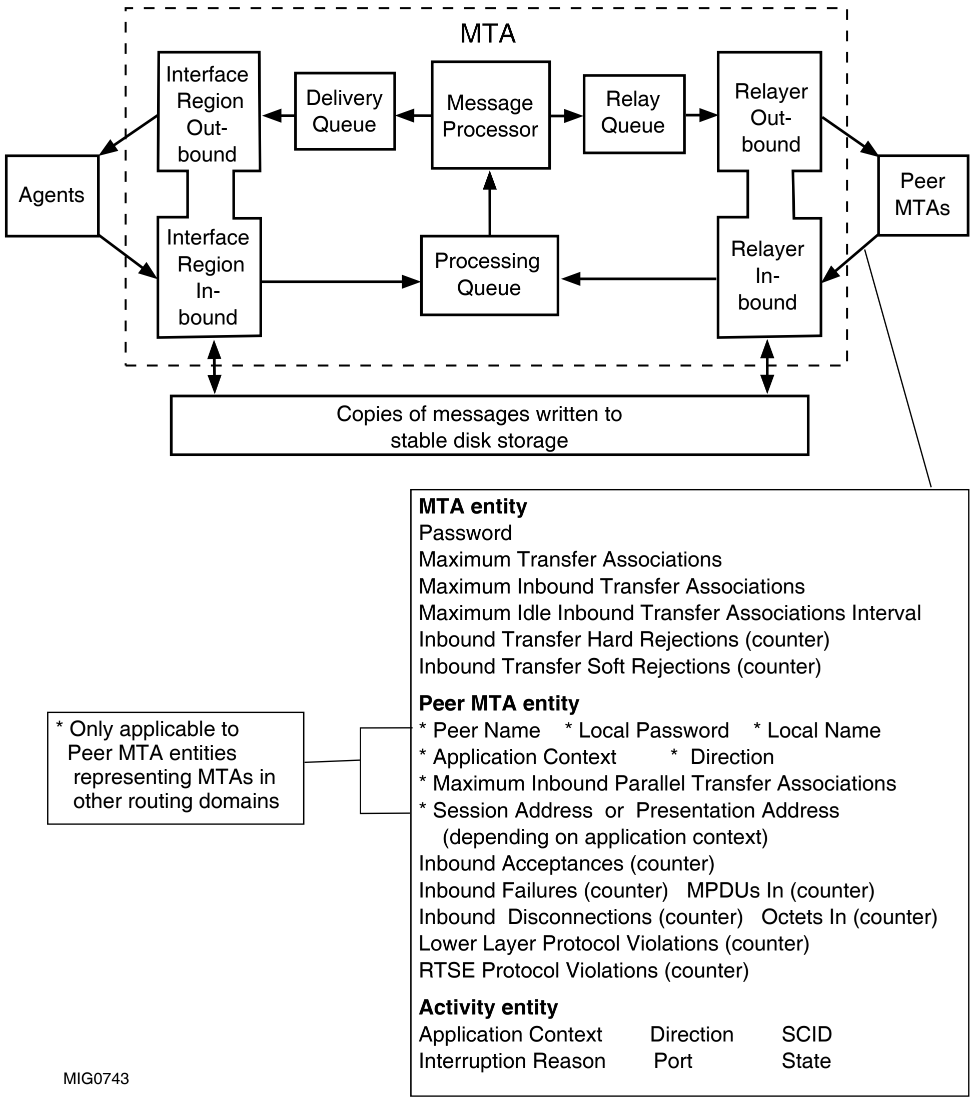
Figure 2–2 also shows counter attributes of the MTA entity and of Peer MTA entities that you can show to monitor inbound associations with peer MTAs.

If there are problems with a connection between two MTAs, or with a received MPDU, the MTA generates appropriate events (see Chapter 16 and Chapter 18).

2.4 Storing Inbound MPDUs

When the MTA receives an MPDU through the Relayer, or through the Interface Region by means of the Shared File interface, the MPDU is already encoded in a standardized form, ASN.1 Basic Encoding Rules (BER), so the MTA writes it to disk unchanged. When the MTA receives an MPDU from the Interface Region through the XAPI interface, the MPDU is not encoded as ASN.1, so the MTA first encodes it before writing it to disk.

Figure 2–2 Management of the Relayer - Inbound


MTA
Interface Region Out-bound
Delivery Queue
Message Processor
Relay Queue
Relayer Out-bound
Agents
Peer MTAs
Interface Region In-bound
Processing Queue
Relayer In-bound
Copies of messages written to stable disk storage
MTA entity
Password
Maximum Transfer Associations
Maximum Inbound Transfer Associations
Maximum Idle Inbound Transfer Associations Interval
Inbound Transfer Hard Rejections (counter)
Inbound Transfer Soft Rejections (counter)
Peer MTA entity
* Peer Name * Local Password * Local Name
* Application Context * Direction
* Maximum Inbound Parallel Transfer Associations
* Session Address or Presentation Address (depending on application context)
Inbound Acceptances (counter)
Inbound Failures (counter) MPDUs In (counter)
Inbound Disconnections (counter) Octets In (counter)
Lower Layer Protocol Violations (counter)
RTSE Protocol Violations (counter)
Activity entity
Application Context Direction SCID
Interruption Reason Port State
* Only applicable to Peer MTA entities representing MTAs in other routing domains
MIG0743

Writing MPDUs to disk provides stable storage in case they need to be recovered, as, for example, after a system failure (see Section 5.1). Because writing to disk can be interrupted, the MTA ensures that the MPDU is completely received and written to disk before accepting responsibility for it and placing a corresponding MPDU in the Processing Queue.

The MTA processes MPDUs in volatile memory. During processing, an MPDU has a different type of encoding from that of the MPDU written to disk. This alternative representation of the MPDU contains only those parts of the MPDU that are necessary for processing. Other parts of the MPDU, such as bodyparts with content, are more efficiently represented by references to the MPDU stored on disk, and are used only when required for operations such as conversion or transfer of the complete MPDU.

When an MPDU has been fully processed and has left the MTA, the MTA deletes the disk copy of that MPDU. The information about the MPDU that is then available to the MTA depends on whether or not the optional Message History logging, Accounting, and Archiving functions have been enabled. If none of these functions are enabled, all record of the MPDU disappears from the MTA.

The exception to this is a **bad message**. This is an MPDU that is recognized as semantically or syntactically incorrect, or that cannot be transferred for some reason other than the unavailability of routing information. The MTA copies such MPDUs to the bad messages directory so that you can examine them later, using the Message Decoder utility provided with the MTA. For information about how to run the Message Decoder, see the appendix describing the operating system specific information.

The MTA automatically creates a Processed Message entity to record information about the current status of all the MPDUs derived from an incoming MPDU as they are processed. If you have enabled Message History logging, the data applicable to Processed Message entities remains in stable storage until automatically purged after an interval that you can set using the Message History Purge Interval attribute of the MTA entity.

3 Message Processing

This chapter describes the operations of the Message Processor, which is responsible for processing MPDUs that it takes from the Processing Queue.

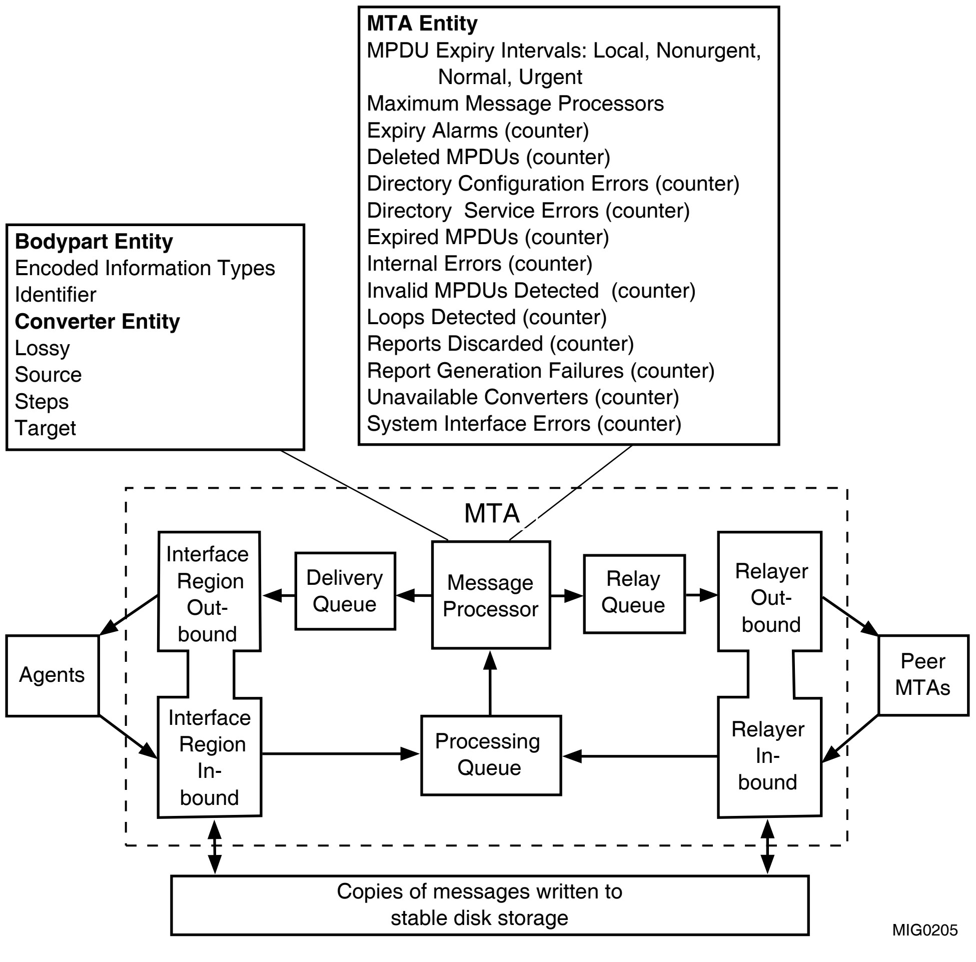
The entity attributes relevant to the management of message processing are shown in Figure 3–1.

3.1 Overview of Message Processing

When the MAILbus 400 MTA has taken an MPDU from the Processing Queue, it uses routing information held in the directory and recipient information contained in the MPDU itself to determine how to route the MPDU and what further processing it must do to the MPDU (see Section 3.2). In addition to routing, when necessary, the Message Processor:

- Expands distribution lists (see Section 3.3)
- Redirects MPDUs (see Section 3.4)
- Checks whether the originator of the MPDU is permitted to send messages to users in a routing domain that is in a different X.400 management domain (see Section 3.5)
- Adds and removes trace information (see Section 3.6)
- Detects Loops (see Section 3.7)
- Checks whether the message format can be accepted by the recipient's User Agent and converts IPM bodyparts (see Section 3.8)
- Downgrades the message content (see Section 3.9)
- Splits MPDUs (see Section 3.10)

Figure 3–1 Management of the MAILbus 400 MTA's Main MPDU Processing Activities


MTA Entity
MPDU Expiry Intervals: Local, Nonurgent, Normal, Urgent
Maximum Message Processors
Expiry Alarms (counter)
Deleted MPDUs (counter)
Directory Configuration Errors (counter)
Directory Service Errors (counter)
Expired MPDUs (counter)
Internal Errors (counter)
Invalid MPDUs Detected (counter)
Loops Detected (counter)
Reports Discarded (counter)
Report Generation Failures (counter)
Unavailable Converters (counter)
System Interface Errors (counter)
Bodypart Entity
Encoded Information Types
Identifier
Converter Entity
Lossy
Source
Steps
Target
MTA
Interface Region Out-bound
Delivery Queue
Message Processor
Relay Queue
Relayer Out-bound
Agents
Peer MTAs
Interface Region In-bound
Processing Queue
Relayer In-bound
Copies of messages written to stable disk storage
MIG0205

The Message Processor can process a number of MPDUs in parallel. You can use the Maximum Message Processors attribute of the MTA entity to specify the maximum number of Message Processors that the MTA can have at any one time.

Increasing the number of Message Processors will increase the throughput of MPDUs, provided that the following conditions are met:

- There are sufficient connections to the Directory System Agent (DSA) available, and the DSA has sufficient capacity.
- There are sufficient resources available on the system on which the MAILbus 400 MTA is running.
- Enough MPDUs pass through the MAILbus 400 MTA to take advantage of parallel processing.

When the Message Processor has finished processing an MPDU, it places the MPDU on the Delivery or Relay Queue. Chapter 4 describes how MPDUs leave the MAILbus 400 MTA through the Interface Region or the Relayer.

3.2 Looking up Addresses for Routing

To understand how the MAILbus 400 MTA uses the directory to look up address information, you need to be familiar with the way in which address and routing information is organized within the directory. The information in the directory is created and maintained using the entities of the MTS module. See Part II of *HP MAILbus 400 MTA Planning and Setup* and the *MTS Module Online Help* for more information about the MTS module.

The MAILbus 400 MTA identifies recipients by their O/R addresses. O/R addresses are stored as entries in the directory, and routing information is stored as attributes of the O/R address entries. Therefore, the first stage of MPDU processing is for the Message Processor to access the directory for information relating to the recipients whose O/R addresses are given in the envelope of the MPDU.

The Message Processor reads the directory for the entry corresponding to the recipient O/R address. If it fails to find such an entry, it ignores the last term from the recipient O/R address, then tries again. It continues to ignore terms and to retry until it either finds a matching O/R address entry or runs out of terms. When it finds a matching address, it checks that the entry contains a routing instruction. If no routing instruction is present, the Message Processor continues the process of ignoring terms and trying to find a matching entry. If it fails to find a matching entry with a routing instruction, it cannot deliver the MPDU for that recipient, and so attempts to send a non-delivery report to the originator of the message.

When the MAILbus 400 MTA receives a message with a recipient O/R address containing a Personal Name attribute but no Common Name attribute, it searches the directory for an O/R address entry with the Personal Name as an attribute. If it finds such an O/R address entry, it routes the message according to the routing instruction in that entry. If the MAILbus 400 MTA does not find such an entry, it routes the message according to the partial O/R address formed as a result of ignoring the Personal Name attribute.

Appendix F gives simple examples of different O/R addresses that contain routing information, and explains how MTAs find the routing information contained in the directory.

3.3 Expanding Distribution Lists

An O/R address can represent an individual or a distribution list containing a number of members. Distribution lists can be nested within other distribution lists. When the Message Processor encounters an O/R address that represents a distribution list, it expands the distribution list, unless the originator has prohibited the expansion of the distribution list. When the Message Processor expands a distribution list, it creates a new MPDU containing only the members of the distribution list as the recipients.

If the expanded distribution list includes the O/R address of any distribution list that has already been processed (including its own O/R address) as a member, the Message Processor does not process those O/R addresses. It processes the other members of the distribution list as previously described.

3.4 Redirecting MPDUs

A recipient can specify a redirection address. This information is held in the directory, in the recipient's O/R address entry. Unless the originator has prohibited redirection, the Message Processor follows any redirections, which replace the original recipient address in the message envelope.

The originator of a message can specify an alternative recipient O/R address in the message envelope. If routing to the intended recipient O/R address fails, the Message Processor attempts to route the MPDU to the alternative recipient O/R address specified in the message envelope. If routing to the alternative recipient succeeds, the Message Processor does not generate a non-delivery report in respect of the intended recipient O/R address. However, if routing to the alternative recipient fails, the Message Processor generates a non-delivery report for the alternative recipient O/R address. This non-delivery report includes information about the redirection of the original MPDU, and is returned to the originator

Note that a non-delivery report cannot be redirected. If an originator has a redirection specified in their own O/R address entry, any non-delivery reports that are destined for this originator are discarded by the MTA and the MTA generates a Report Discarded event.

3.5 Sending Messages Between Different X.400 Management Domains

When the Message Processor receives a message or probe addressed to a recipient in another routing domain, the Message Processor checks the Different CCITT Domain attribute of the Domain entity in the directory that represents the other routing domain. The Domain entity is an entity of the MTS module. The Different CCITT Domain attribute is used to indicate whether or not the routing domain represented by the Domain entity is part of an X.400 management domain that is different from your X.400 management domain.

If a recipient's routing domain is in a different X.400 management domain, (that is, if the Different CCITT Domain attribute is set to TRUE) the Message Processor checks the May Cross CCITT Boundaries attribute of the user's O/R address entry in the directory. The May Cross CCITT Boundaries attribute indicates whether a user with this O/R address is permitted to send messages to users in routing domains that are part of an X.400 management domain that is different from your X.400 management domain. In other words, this attribute indicates whether the user is permitted to send messages that cross X.400 management domain boundaries.

If the originator is permitted to send messages that cross X.400 management domain boundaries, the Message Processor processes the message so that it can be transferred to the recipient's routing domain. If the originator is not permitted to send messages that cross X.400 management domain boundaries, the Message Processor does not process the message for transfer to the recipient's routing domain, but instead sends a non-delivery report to the originator.

See the *MTS Module Online Help* for more information about ORaddress and Domain entities.

3.6 Adding and Removing Trace Information

There are two types of trace information that are recorded in the message envelope:

- External trace information

  External trace information identifies each X.400 management domain that has transferred the message.

- Internal trace information

  Internal trace information identifies each MTA that has transferred the message within an X.400 management domain.

The Message Processor is responsible for adding and removing trace information. Section 3.6.1 describes when the Message Processor adds external trace information and Section 3.6.2 describes when the Message Processor adds and removes internal trace information.

3.6.1 Adding External Trace Information

The Message Processor uses the GDI information specified by the Global Domain Identifiers attribute of the MTA entity in the MTS module as the external trace information. If there is a list of GDIs in the Global Domain Identifiers attribute, that is, the routing domain is multi-homed, the Message Processor uses the first GDI in the list as external trace information.

The Message Processor adds external trace information to messages in two circumstances:

- When it receives a message that has been submitted from a User Agent or Message Store.
- When it detects that a message originated from another X.400 management domain, as indicated by the Different CCITT Domain attribute of the relevant Domain entity.

The Message Processor does not remove external trace information from messages.

3.6.2 Adding and Removing Internal Trace Information

The MTA uses the MTA name (including area name(s)) as the internal trace information.

The Message Processor adds internal trace information to the envelope of all messages it processes. It removes internal trace information when it detects that the message will be transferred to another X.400 management domain, as indicated by the Different CCITT Domain attribute of the relevant Domain entity.

3.7 Detecting Loops

The MTA uses trace information to detect loops in the route taken by a message. When the Message Processor receives a message, it checks the GDIs in the external trace information and the number of MTAs in the internal trace information. The Message Processor detects an external trace loop when it receives a message that has previously been relayed to the X.400 management domain of which the MTA is a part. The Message Processor detects an internal trace loop when it receives a message that has been handled by 50 MTAs within the same X.400 management domain.

When the Message Processor detects a loop, it does not transfer the message, and instead sends a non-delivery report to the originator. The Message Processor copies the message that has been sent through a loop to the bad messages directory.

3.8 Checking Message Format and Converting IPM Bodyparts

The envelope of an MPDU specifies the MPDU's content type (for example, IPMS, EDI, Undefined) and, where relevant, the constituent data formats or bodyparts.

When the Message Processor receives a message, it checks its content type, content length, and the constituent data formats or bodyparts, against content information stored in the directory. The content information in the directory is held in a recipient's O/R address entry. The Message Processor checks the O/R address entry for each recipient that is specified in the message envelope. Using the content information the Message Processor decides what actions to perform for each message recipient. If the content type or the content length of the message are not acceptable to a recipient, the Message Processor does not deliver the message and returns a non-delivery report to the originator.

If the content type of an MPDU is IPMS, and IPMS content type is acceptable to the message recipients, the Message Processor determines whether the bodyparts of the MPDU are also acceptable or whether any of the bodyparts must be converted. The Message Processor determines which bodyparts are acceptable to each recipient from the content information in the recipient's O/R address entry in the directory.

Note that the MAILbus 400 MTA attempts to convert only IPMS content types. Messages containing content types other than IPMS are either transferred without conversion, or not delivered. When the Message Processor does not deliver a message, it returns a non-delivery report to the originator.

IPM bodypart converters enable the MAILbus 400 MTA to convert the constituent bodyparts of IPM contents into other bodyparts. A converter is a software tool that translates data from one encoding format or character set to another. An example of a conversion requirement is when an IPM containing a bodypart in Teletex arrives at the MAILbus 400 MTA, but the User Agent of one of the recipients can only handle IA5 text. In this situation, the Teletex bodypart needs to be converted to IA5 text for that recipient.

3.8.1 Criteria for Conversion

The MAILbus 400 MTA converts the bodyparts of IPM contents according to the following criteria:

- The instructions contained in the MPDU envelope

  The MPDU envelope can contain instructions set by the originator as to whether or not conversions are permitted. If conversions are permitted, the envelope of the MPDU indicates whether any loss of data is permitted during these conversions.

- The types of bodypart acceptable to recipients

  These are indicated by the Content Information attribute of the recipient's O/R address entry stored in the directory.

- The converters available to the MTA

  Section 3.8.2 describes the types of IPM bodypart converters available.

If the content type of an MPDU is IPMS, and the MPDU includes bodyparts that are not acceptable to a recipient, the MAILbus 400 MTA uses the relevant converter, or converter sequence, to convert each bodypart as necessary:

- If a bodypart is acceptable to a recipient, the MTA transfers it without conversion.
- If a bodypart is not acceptable to a recipient and the MTA can convert it to an acceptable IPM bodypart, the Message Processor performs the most acceptable conversion it can, based on the recipient's preferences specified in the O/R address entry.
- If a bodypart is not acceptable to a recipient and the Message Processor cannot access an appropriate converter, the bodypart is transferred without conversion, unless this is the last MTA in the route of the MPDU.

  If the MPDU is at the MTA which is to deliver or export the MPDU, the MTA delivers, exports, or does not deliver the MPDU according to the content information specified in the recipient's O/R address entry. If the content information includes a list of preferred bodyparts, and that list includes a value signifying that *any* bodypart is acceptable, the MTA delivers or exports the MPDU without conversion. Otherwise, the MTA does not deliver the MPDU and attempts to send a non-delivery report to the message originator.

To maximize the probability of successful transfer, MPDUs are converted as early as possible during routing; that is, by the first MTA in the routing domain that receives the MPDU and has an appropriate IPM bodypart converter available.

For more information about converting IPM bodyparts, see Chapter 11.

3.8.2 Converters Available to the MTA

A set of IPM bodypart converters for common messaging formats (such as IA5, T.61 (Teletex) and DDIF) is supplied with the MAILbus 400 MTA. A set of corresponding Converter and Bodypart entities is created as part of the MTA startup procedure. For the MAILbus 400 MTA to convert IPM bodyparts, these entities must exist. Therefore, if you modify the part of the MTA startup procedure that creates these entities, or if you delete any of these entities, you prevent particular conversions.

You can write or obtain additional IPM bodypart converters, and define new IPM bodypart types. This means that the MTA can do a potentially unlimited range of IPM bodypart conversions. If you install a new converter or define a new IPM bodypart type, you must use MTA management commands to create corresponding Converter and Bodypart entities (see Chapter 11).

Where there is no converter available to convert one bodypart directly to another, the MAILbus 400 MTA can use a number of converters in sequence to complete the conversion. See Section 12.2.7 for details of how to do this.

3.9 Downgrading Message Content

The interpersonal messaging (IPMS) content type is defined in both the 1984 and 1992 MHS Standardss. If the information in the directory indicates that the recipient can only accept message contents of the type Interpersonal messaging 1984, the Message Processor downgrades the message content. Downgrading the message content is described in Chapter 6. Downgrade is invoked only when all other requested conversions are complete.

3.10 Splitting MPDUs

When an MPDU is sent to more than one recipient, or an MPDU is converted, it must eventually be split into as many occurrences as there are destinations. To maximize routing efficiency, an MPDU is split as late as possible in its route, thus avoiding unnecessary copies of MPDUs being transferred over a network. An MPDU is not split until it reaches an MTA where there are recipients that cannot be reached by a common route, or where the content has been converted or downgraded in different ways for different recipients.

The Message Processor always splits an MPDU into the minimum possible number of related MPDUs consistent with available routing possibilities and the requirements of recipients. Having split the MPDU, the Message Processor places each resultant MPDU on a queue. If an MPDU is for delivery or export to an Agent, the Message Processor sends it to the Delivery Queue.

If the MPDU is for one or more recipients at another MTA, the Message Processor sends it to the Relay Queue for transfer to the next MTA in the MPDU's route.

MPDUs representing reports are never split, because, by definition, they have only a single destination.

4 How MPDUs Leave the MTA

This chapter describes the different ways in which MPDUs leave the MTA for forwarding to one or more destinations in an MHS.

4.1 Types of Exit from the MTA

An MPDU leaves the MTA by one of the following routes:

- Delivery

  This is where the MPDU, representing a message or report, leaves through the Interface Region of the MTA and is delivered to a User Agent or Message Store (see Section 4.2).

- Export

  This is where an MPDU, representing a message, probe, or report, leaves the MTA through the Interface Region and is transferred through a Gateway to another messaging system (see Section 4.2). An exported MPDU leaves the MAILbus 400 MTA, but it does not necessarily leave the MTS.

- Outbound transfer

  This is where the MPDU leaves the MTA through the Relayer and is transferred to another X.400 MTA in your routing domain or in another routing domain (see Section 4.3).

4.2 MPDUs Leaving Through the Interface Region

When the MTA has an MPDU for an Agent, the Interface Region takes the MPDU from the Delivery Queue and makes it available for either delivery or export.

The MPDUs in the Delivery Queue are maintained in the order of their priority and arrival time. The Delivery Queue is headed by the MPDUs of the highest priority. For MPDUs of equal priority, those that have waited at the MTA the longest are at the front of the queue. The MPDU currently at the front of the queue is the first that the MTA passes to the Agent when the Agent connects to the MTA to collect its messages. The Interface Region can process outbound MPDUs for several Agents at the same time.

A registered Agent of the MTA can use either the XAPI interface, or the Shared File interface. The Shared File interface is only used by Gateways.

An unregistered Agent of the MTA can only use the XAPI interface.

4.2.1 Registered Agents Using the XAPI Interface

For registered Agents using the XAPI interface, the Interface Region does the following:

1. Activates, if possible, the appropriate Agent application

   If the Agent is based on the 1984 MHS Standards, the Interface Region downgrades the message envelope (see Chapter 6).

   The Interface Region tries to make known the availability of an MPDU and bring the application into action to fetch the MPDU. You can use the Invocation Filename attribute of the Agent entity to specify a shell script or image (Tru64 UNIX), or command procedure (OpenVMS), to be run in order to start up the Agent.

   The Interface Region tries to run the invocation file, but to avoid over-frequent communication with any particular Agent, it does not try more than once for any particular MPDU. In addition, the Interface Region will never run the invocation file for the same Agent more than once within a minute. If there is an existing connection to a particular Agent, the invocation file for the Agent is not used.

2. Holds the MPDU in the Delivery Queue until it is collected or expires

   Whether or not the Agent has been notified of the availability of an MPDU, it must initiate a connection to the MTA in order to collect any available MPDUs from the Delivery Queue. The Agent can be permanently connected, or make a connection only infrequently. If the MPDU is not collected from the Delivery Queue, it eventually expires. In this case, the MTA attempts to send a non-delivery report to the originator of the message and deletes the MPDU from the Delivery Queue.

4.2.2 Unregistered Agents Using the XAPI Interface

For unregistered Agents using the XAPI interface, the Interface Region cannot use an Invocation File and only performs the actions described in step 2 of Section 4.2.1.

4.2.3 Registered Agents Using the Shared File Interface

For registered Agents using the Shared File 1984 interface, the Interface Region downgrades the message envelope and writes the MPDU to a file in the Output Queue, which is then accessed by the Agent (see Chapter 6). For registered Agents using the Shared File 1992 interface, the Interface Region writes the MPDU to a file in the Output Queue, which is then accessed by the Agent.

Thereafter, the Interface Region treats the MPDU as having been exported, and the Agent is responsible for any further processing.

4.2.4 All Agents

When an MPDU has been written to a file for an Agent using the Shared File interface, or has been collected by an Agent using the XAPI interface, the Interface Region does the following:

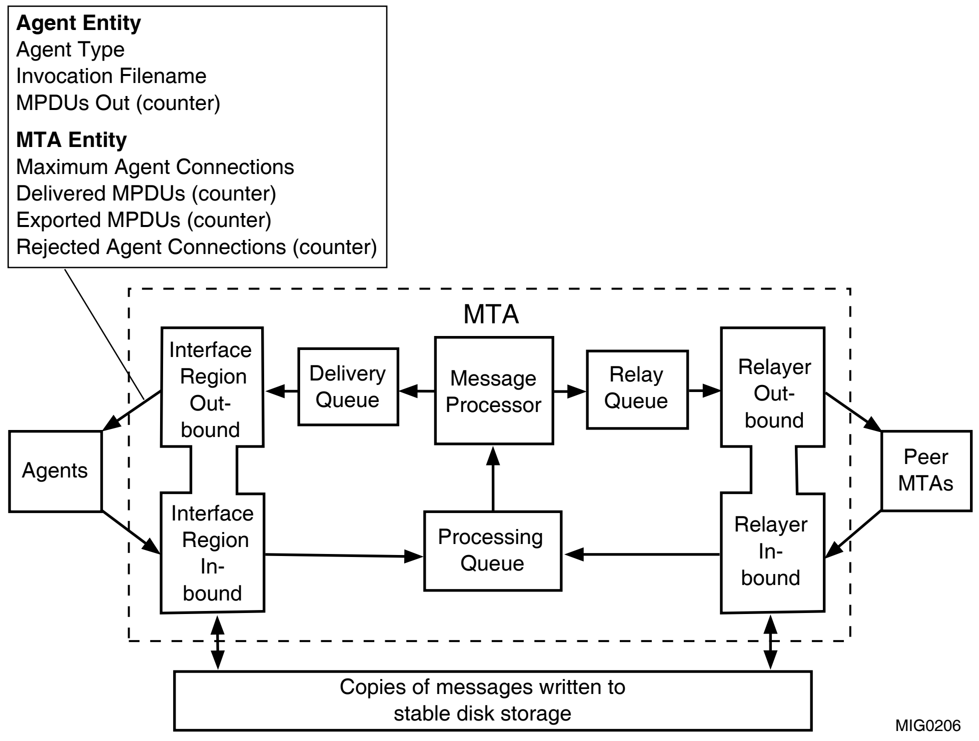
- Ends its responsibility for the MPDU
- Writes a journal record for the MPDU
- Deletes the original version of the MPDU from stable storage on disk (see Section 2.4)

The Interface Region generates reports in respect of MPDUs that it cannot deliver. It then places these reports on the Processing Queue for delivery to the originator of the message.

Outbound communications with Agents are subject to conditions set by the values of the characteristic attributes of Agent entities and the MTA entity shown in Figure 4–1. These may be values supplied with the product, or values you set when tuning the MTA. Figure 4–1 also shows counter attributes that you can show to monitor outbound communications with Agents.

If there are problems with a connection to an Agent, the MTA generates appropriate events (see Chapter 19).

Figure 4–1 Management of the Interface Region - Outbound



4.3 MPDUs Leaving Through the Relayer

When the MTA has an MPDU for a peer MTA, the Relayer takes it from the Relay Queue and does the following:

1. Seeks or initiates an outbound association

   If the MTA has an existing idle association to the appropriate peer MTA, it uses this association to transfer the MPDU. If no idle association is available, the MTA attempts to create a new association. (Section 7.3.4 describes the conditions under which the MTA is unable to create a new association to a peer MTA.)

If a new association can be initiated, the Relayer operates the RTSE protocol to establish communication with the peer MTA. The type of X.400 connection that the Relayer uses depends on whether or not the peer MTA is in the same routing domain. If it is in the same routing domain, the Relayer uses OSI ACSE associations. If the peer MTA is in a different routing domain, the type of connection that the Relayer uses is determined by the Application Context attribute of the appropriate Peer MTA entity. This attribute can have one of the following values:

- MTS Transfer

  This indicates that the peer MTA fully supports 1992 MHS Standards, and uses OSI ACSE associations.

- MTS Transfer Protocol

  This indicates that the peer MTA accepts 1992 X.400 messages, using OSI Session connections.

- MTS Transfer Protocol 1984

  This indicates that the peer MTA accepts only 1984 X.400 messages, using OSI Session connections. In this case, the boundary MTA downgrades messages before it transfers them to the peer MTA (see Chapter 6).

2. Transfers the MPDU

   When the Relayer starts to transfer an MPDU, it creates an Activity entity, which it uses to provide information about the state of the association and its activity in transferring the MPDU. (See Section 1.3.4 for information about Activity entities.)

   While transferring an MPDU, the Relayer keeps checkpoint information about the progress of the transfer. This information can be used in case of a need to recover the association and resume the transfer of the MPDU.

3. Queues the MPDU for retry if transfer fails or no association could be established

   It can happen that the Relayer fails to establish an association or fails to complete the transfer of an MPDU to a peer MTA. This could be because the peer MTA is not operational and so cannot be reached, or because a communications failure causes the peer MTA to become unavailable before the transfer of the MPDU is complete. In such instances, the Relayer tries again to transfer the MPDU. If this fails the Relayer takes the following action according to reason for failure;

- If the Relayer has failed to establish an association, the Relayer continues trying to transfer the MPDU at intervals, as described in Section 4.3.2, until the MPDU expires.
- If the Relayer fails to complete the transfer of an MPDU, the Relayer attempts to transfer the MPDU at intervals, as described in Section 4.3.2.

  If the transfer fails to complete three times, the Relayer discards any recovery information and attempts to transfer the MPDU at one minute intervals, three more times. If transfer still fails, the Relayer issues an Invalid MPDU event and sends a non-delivery report to the originator.

4. Ends its responsibility for the MPDU

   Once the MPDU has been successfully transferred and the peer MTA has accepted responsibility for it, the Relayer writes a journal record for the MPDU.

5. Deletes the MPDU

   After the Relayer has ended its responsibility for the MPDU, it deletes the MPDU from volatile memory and, if no other MPDUs refer to it, deletes the original version of the MPDU from stable storage on disk (see Section 2.4).

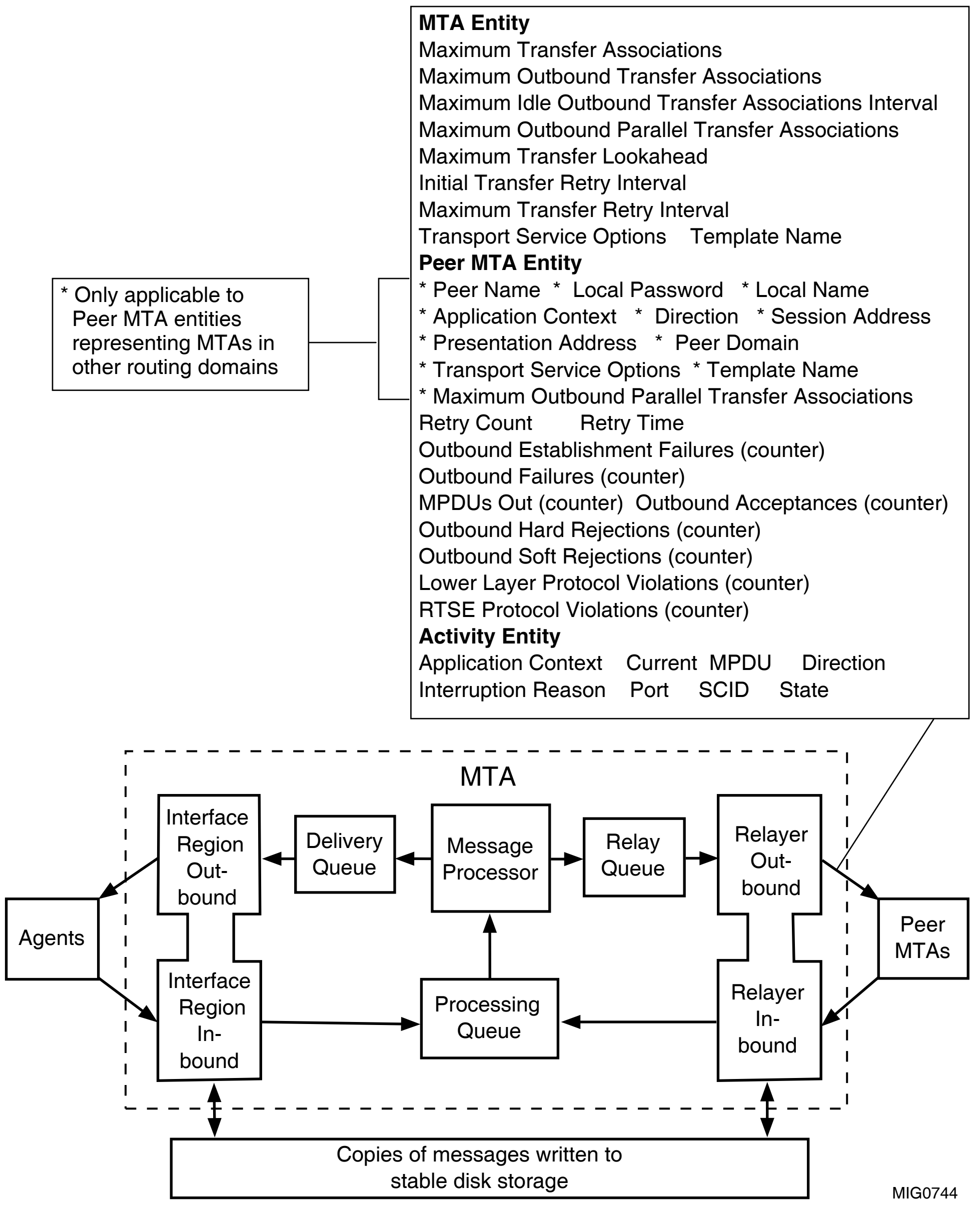
Outbound associations with peer MTAs are subject to conditions set by the values of the characteristic attributes of the MTA entity and Peer MTA entities shown in Figure 4–2. These may be values that are supplied with the MTA or those that you set when tuning the MTA. Figure 4–2 also shows counter attributes that let you monitor outbound associations with peer MTAs.

If there are problems with a connection between two MTAs, the MTA generates appropriate events (see Chapter 16).

4.3.1 Selecting MPDUs from the Relay Queue

The MTA maintains MPDUs in the Relay Queue in the order of their priority and arrival time. The Relay Queue is headed by MPDUs of the highest priority. For MPDUs of equal priority, those that arrive at the MTA first are at the front of the queue. The MPDU currently at the front of the queue is normally the first to be considered for transfer by the Relayer.

Figure 4–2 Management of the Relayer - Outbound


MTA Entity
Maximum Transfer Associations
Maximum Outbound Transfer Associations
Maximum Idle Outbound Transfer Associations Interval
Maximum Outbound Parallel Transfer Associations
Maximum Transfer Lookahead
Initial Transfer Retry Interval
Maximum Transfer Retry Interval
Transport Service Options Template Name
Peer MTA Entity
* Peer Name * Local Password * Local Name
* Application Context * Direction * Session Address
* Presentation Address * Peer Domain
* Transport Service Options * Template Name
* Maximum Outbound Parallel Transfer Associations
Retry Count Retry Time
Outbound Establishment Failures (counter)
Outbound Failures (counter)
MPDUs Out (counter) Outbound Acceptances (counter)
Outbound Hard Rejections (counter)
Outbound Soft Rejections (counter)
Lower Layer Protocol Violations (counter)
RTSE Protocol Violations (counter)
Activity Entity
Application Context Current MPDU Direction
Interruption Reason Port SCID State
* Only applicable to Peer MTA entities representing MTAs in other routing domains
MTA
Interface Region Out-bound
Delivery Queue
Message Processor
Relay Queue
Relayer Out-bound
Agents
Peer MTAs
Interface Region In-bound
Processing Queue
Relayer In-bound
Copies of messages written to stable disk storage
MIG0744

Strict adherence to the principle of choosing MPDUs for transfer does not make optimum use of established associations. Sometimes the Relayer has an idle association that is unsuitable for the MPDU at the head of the queue but which could be used for an MPDU later in the queue. You can set a value for the Maximum Transfer Lookahead attribute of the MTA to allow the Relayer to select MPDUs that are not at the head of the Relay Queue, in order to take advantage of existing associations. The value of this attribute determines how far down the queue the Relayer can look in order to select an MPDU for transfer across an existing association.

4.3.2 Outbound Transfer Retries

When a peer MTA cannot be reached, or a transfer fails to complete, the Relayer establishes a retry set for MPDUs destined for the unavailable peer MTA. It also starts a timer associated with the retry set, with a value derived from the Initial Transfer Retry Interval attribute of the MTA entity. Whenever the Relayer selects an MPDU from the Relay Queue to begin a transfer, it checks whether a retry set exists for the MTA to which the MPDU is to be sent. If a retry set exists for that MTA, the Relayer adds the MPDU to the set. If not, it tries to transfer the MPDU.

When the timer associated with a retry set expires, all the MPDUs in the set are placed on the Relay Queue again. If transfer of the first MPDU of the retry set is successful, the Relayer transfers the other MPDUs in the usual way. If transfer of the first MPDU of the retry set fails, the retry timer is restarted, with double the initial retry interval, and MPDUs again accumulate in the retry set. The Relayer maintains the retry set and retries the MPDUs until it successfully transfers them to the peer MTA or until the MPDUs eventually expire.

Each time the Relayer restarts the timer for a retry set for a particular unavailable peer MTA, the Relayer doubles the transfer retry interval. When the retry interval reaches the value of the Maximum Transfer Retry Interval attribute of the MTA entity, or if it would exceed this value when doubled again, the Relayer stops increasing the interval between retries. Eventually, the MPDU expires; the MTA attempts to send a non-delivery report to the originator of the message and deletes the MPDU from the Relay Queue. For information about modifying to the retry interval, see Section 7.3.5 and Table 7–4.

5 Keeping Records of MPDUs

The MAILbus 400 MTA has several methods of keeping records of its activities in handling MPDUs. This chapter gives an overview of:

- Journaling

  Journaling enables the MAILbus 400 MTA to keep track of the transfer status of all MPDUs for which it is currently responsible (see Section 5.1.1). The MTA uses Journaling to reconstruct and transfer MPDUs whose processing or transfer was interrupted by system failure or disabling of an MTA.

- Accounting

  This optionally-enabled function allows the MAILbus 400 MTA to record information about its own MPDU traffic (see Section 5.2). It is useful for managing or costing the work load of the MTA. Accounting enables you to choose the details of information to be recorded.

- Archiving

  This optionally-enabled function allows the MAILbus 400 MTA to put complete copies of selected MPDUs into the Archive directory (see Section 5.3). Archiving enables you to choose the registered Agents or peer MTAs for which all MPDUs are archived. You can also choose whether to archive incoming or outgoing MPDUs, or both.

- Message History logging

  This optionally-enabled function allows the MAILbus 400 MTA to record historical data about all the MPDUs representing messages (not probes or reports) that it has processed (see Section 5.4). It is useful for tracing messages when solving MTS problems, and is most useful when enabled at the boundary MTAs in an MTS.

Figure 5–1 Management of Accounting, Archiving, and History Logging

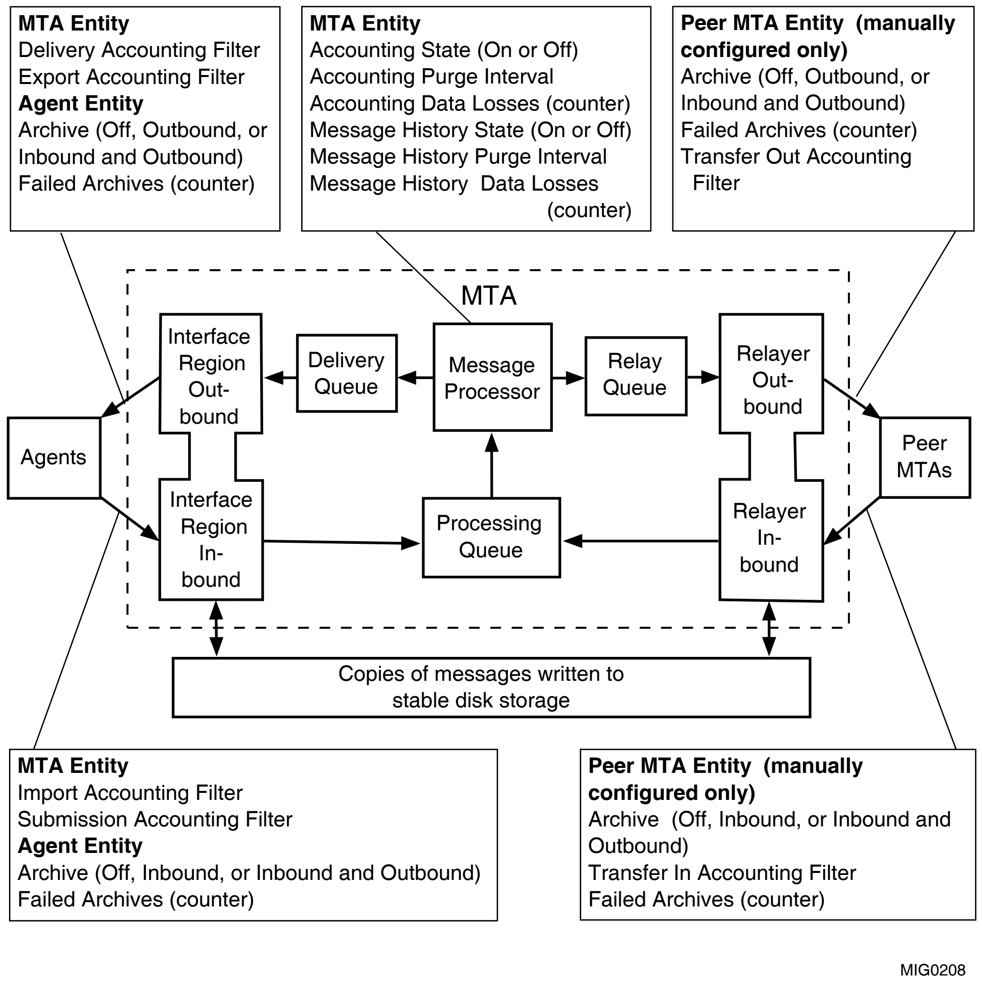

MTA Entity
Delivery Accounting Filter
Export Accounting Filter
Agent Entity
Archive (Off, Outbound, or Inbound and Outbound)
Failed Archives (counter)
MTA Entity
Accounting State (On or Off)
Accounting Purge Interval
Accounting Data Losses (counter)
Message History State (On or Off)
Message History Purge Interval
Message History Data Losses (counter)
Peer MTA Entity (manually configured only)
Archive (Off, Outbound, or Inbound and Outbound)
Failed Archives (counter)
Transfer Out Accounting Filter
MTA
Interface Region Out-bound
Delivery Queue
Message Processor
Relay Queue
Relayer Out-bound
Agents
Peer MTAs
Interface Region In-bound
Processing Queue
Relayer In-bound
Copies of messages written to stable disk storage
MTA Entity
Import Accounting Filter
Submission Accounting Filter
Agent Entity
Archive (Off, Inbound, or Inbound and Outbound)
Failed Archives (counter)
Peer MTA Entity (manually configured only)
Archive (Off, Inbound, or Inbound and Outbound)
Transfer In Accounting Filter
Failed Archives (counter)
MIG0208

Figure 5–1 provides an overview of the entity attributes that you use to control and monitor the Accounting, Archiving, and Message History logging functions of the MAILbus 400 MTA.

5.1 Journaling and Recovery

The MTA stores MPDUs for which it is responsible in stable disk storage. In this way, the messages can survive a system failure. If the system that the MTA is running on fails and is restarted, the MTA automatically recovers from the system failure and continues the processing and transfer of MPDUs whose processing or transfer was interrupted by the system failure.

5.1.1 Journaling

MPDUs are frequently split into a number of MPDUs. This means that a system failure can affect some MPDUs, split from a single MPDU, while they are still in the course of processing, delivery, or transfer. Only those MPDUs that have not been completely transferred need to be recovered. If the disk copy of an MPDU were to be processed again for every recipient, any recipient to whom a copy of the MPDU had already been transferred would receive a duplicate. To minimize such duplication, the MTA maintains, on disk, an active journal file of completed operations. Journal records are appended to the journal under the following circumstances:

- When another MTA acknowledges the transfer of an MPDU representing a message, probe, or report.
- When an Agent acknowledges the delivery of a message or a report.
- When the MTA discards a report that it has generated from a message because it cannot route the report.

The MTA automatically purges the journal file when it reaches a certain size. It determines this size on the basis of previous MPDU traffic. This purging releases disk space and reduces the time needed to process the journal file during recovery.

The MTA purges the journal file by a rollover process which involves the following:

1. Creating a new journal file.
2. Copying journal records from the old journal file to the new one.

   This happens if the MTA has processed only part of the MPDUs that were split from a single MPDU in stable disk storage. The MTA copies the journal records for those split MPDUs that were processed completely at the time of journal rollover.

3. Deleting the old journal file (or files, if the MTA failed during a previous journal rollover).

The MTA continues to function during rollover. Rollover cannot take place while recovery is in progress (see Section 5.1.2).

5.1.2 Recovery

The MTA uses its journal file to recover from a system failure or reboot, or when it is enabled after having been disabled. During recovery, the MTA uses the journal file, in conjunction with the disk copy of MPDUs the MTA has responsibility for, to recover MPDUs for destinations that have not yet received them.

The MTA first determines whether there is more than one journal file. This could occur if a system failed during journal rollover, and each of the files could contain valid records. In this case, the MTA processes all files.

The MTA then reads, from disk, the MPDUs for which it is currently responsible, and determines whether or not there are relevant journal file records. If the journal file does not contain a record for every recipient of an MPDU, the MPDU was not completely processed. The MTA annotates incompletely processed MPDUs so that they are put into the Processing Queue in order to complete processing for the remaining recipients. Meanwhile, normal operation of the MTA begins and new MPDUs are accepted and processed in parallel with the recovery operation, including the writing of new journal records to the journal file.

If, for any reason, the MAILbus 400 MTA is stopped, a small number of journal records may be lost. The effect of this is that a few recipients of an MPDU receive an extra copy of it after MTA recovery.

Because recovery involves a step-by-step resumption of normal processing, including journaling, it does not matter if a system failure occurs during MPDU recovery.

In addition to an MTA recovering its own messages, it can also recover messages from a peer MTA that is in the same routing domain. You might want an MTA to recover a peer MTA's messages when the peer MTA is unavailable for some time, for example, because the system that the peer MTA is running on is shut down for maintenance. An MTA recovers messages from a peer MTA by copying them from the peer MTA and subsequently deleting them from the peer MTA's disk. As a result, the MTA takes responsibility for a message it copies from the peer MTA and attempts to process the message in the same way as the peer MTA would have done. For more information about using an MTA to recover messages from a peer MTA see Section 18.8.

5.2 Accounting

The MTA can record data about the MPDUs it processes. You can use Accounting data to determine MPDU traffic in and out of an MTA for the purpose of billing MTS users, or for statistical analysis of aspects of the MTA work load.

To use Accounting, you must first switch it on by means of the Accounting State attribute of the MTA entity. Figure 5–1 shows where Accounting information is captured during MPDU transfer through the MTA. Note that Accounting information relating to communications with peer MTAs applies only to peer MTAs in other routing domains.

There are Accounting filters that you can set to choose the data items that are written to the Accounting log. See Chapter 8 for details of Accounting filters. The data items written to the Accounting log include, for each MPDU, the originator, recipients, priority, and size. If Accounting is switched on and an item of Accounting data cannot be logged, the MTA generates an event and increments a corresponding counter.

Accounting data is held on disk and is retained for a time period that you set with the Accounting Purge Interval attribute of the MTA entity. The MTA writes Accounting data to disk in ASN.1 (BER). The MTA includes an Accounting Decoder tool for converting this data to text in a file which can be used to generate invoices or for analysis.

For information on management of Accounting files see Chapter 8. For information about how to use the Accounting Decoder tool see the appendix describing the operating system specific information.

5.3 Archiving

The MTA can retain, on disk, complete copies of MPDUs it processes. To use Archiving, you must first switch it on by means of the Archive attributes of the Agent and Peer MTA entities for which you require MPDUs to be kept. Archiving can be used for registered Agents, and for peer MTAs in other routing domains. You can set Archiving to operate for inbound or outbound communications, or both. Figure 5–1 shows the stages in MPDU transfer through the MTA at which Archive data is captured.

Archive copies of MPDUs are stored in ASN.1 (BER) format. The MAILbus 400 MTA includes a Message Decoder tool which you can use to decode archived messages in order to produce a readable copy of them.

The MTA does not automatically purge archived files. If you do not regularly transfer files from the disk to some other storage medium, the archive area becomes full and Archiving stops working. If the MTA fails to archive an MPDU, it generates an event and increments the corresponding counter.

For information about the management of archived files see Chapter 9. For information about how to use the Message Decoder tool see the appendix describing the operating system specific information.

5.4 Message History Logging

The MTA automatically creates a Processed Message entity for every message that enters the MTA. It then updates the Recipient Information status attribute of the Processed Message entity for each MPDU derived from the incoming message as it passes through the MTA (see Figure 5–2).

The MTA can retain Message History information on all MPDUs that represent messages (not probes or reports) for which it has taken responsibility. This enables you to trace what happened to a particular MPDU at the MTA. If Message History logging is switched on at other MTAs in the MTS, you can also trace the MPDU through other MTAs (see Chapter 10).

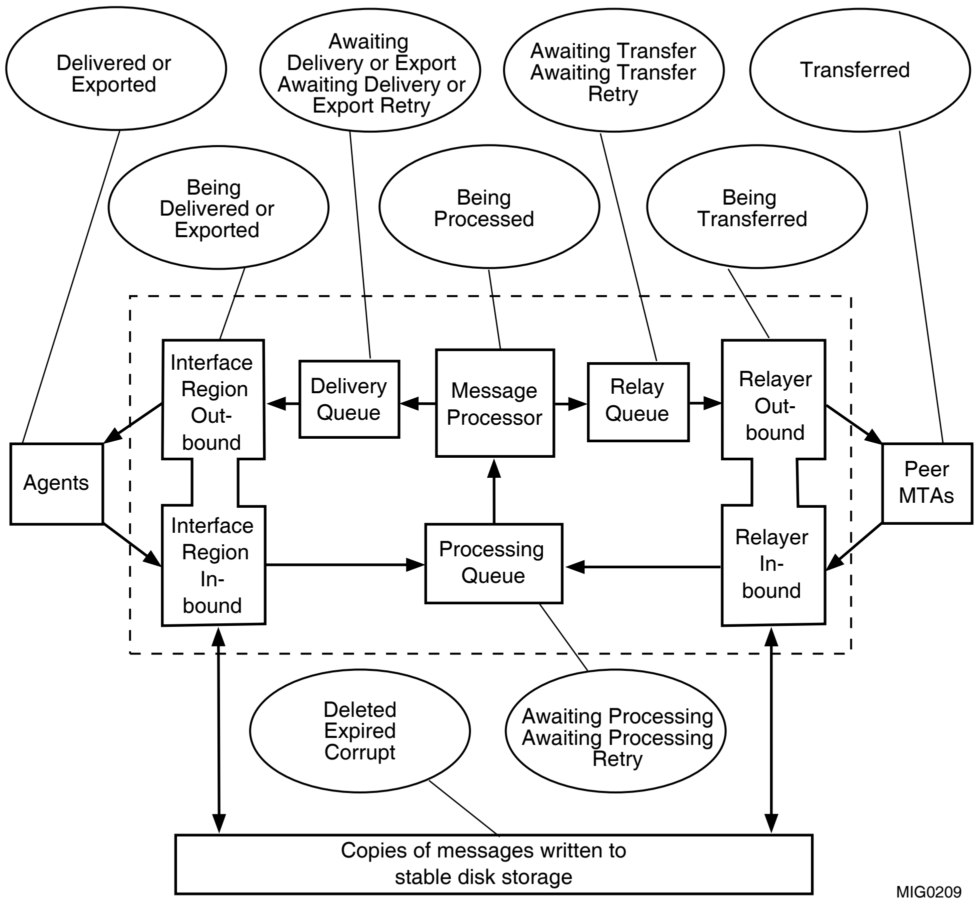
The MTA retains Message History information only if Message History logging is switched on. To switch on Message History logging, you must enable it by means of the Message History State attribute of the MTA entity (see Figure 5–1). Message History information is held on disk and is retained for a time period that you can set by using the Message History Purge Interval attribute of the MTA entity.

If Message History logging is switched on and fails for a particular MPDU, the MTA generates an event and increments the corresponding counter.

5.5 Bad Messages

A bad message is any MPDU which the MTA recognizes as semantically or syntactically incorrect, or which it cannot deliver or transfer for some reason other than the unavailability of routing information. The MTA copies such MPDUs to a file in a directory assigned to bad messages. You can examine bad messages by means of the Message Decoder tool, provided with the MTA, as described in the appendix describing the operating system specific information.

Figure 5–2 Processed Message Entity - MPDU State Values


Delivered or Exported
Awaiting Delivery or Export Awaiting Delivery or Export Retry
Awaiting Transfer Awaiting Transfer Retry
Transferred
Being Delivered or Exported
Being Processed
Being Transferred
Interface Region Out-bound
Delivery Queue
Message Processor
Relay Queue
Relayer Out-bound
Agents
Peer MTAs
Interface Region In-bound
Processing Queue
Relayer In-bound
Deleted Expired Corrupt
Awaiting Processing Awaiting Processing Retry
Copies of messages written to stable disk storage
MIG0209

6 Downgrading

A MAILbus 400 MTA can interwork with a peer MTA, or an Agent, that conforms to the 1984 MHS Standards. The 1992 MHS Standards define rules for downgrading a message so that it is compatible with the CCITT 1984 X.400 Series of Recommendations. The MAILbus 400 MTA implements these rules. In addition, it implements extra procedures, not defined in the Standards, to facilitate interworking with a 1984 X.400 MHS. These procedures are concerned with the use of the Common Name O/R address attribute and the use of the IPMS content type. This chapter describes:

- How the MAILbus 400 MTA downgrades messages (Section 6.1).
- When the MAILbus 400 MTA downgrades (Section 6.2 and Section 6.3).
- How to set up your MTS to support downgrading (Section 6.4).

6.1 How the MAILbus 400 MTA Downgrades Messages

The following sections describe how the MAILbus 400 MTA downgrades the envelope, the O/R address attributes, and the content of a message.

To fully understand these sections, you need an understanding of the MHS standards. You may also need to refer to copies of the standards and their corresponding profiles. See Appendix A for information about how to obtain these standards and profiles.

6.1.1 Downgrading the Message Envelope

The MAILbus 400 MTA implements the following downgrading procedures for the message envelope:

- Those defined in:
  - Annex B of CCITT Recommendation X.419
  - International Standard ISO/IEC 10021-6

- Those defined in:
  - Annex D of ISO/IEC ISP 10611-1 (1994)
  - Chapter 8 of the OIW Stable Implementation Agreements, December 1993
  - CEN/CENELEC profile ENV 41214
- Those defined in RFC 1328 *X.400 1988 to 1984 Downgrading*

The way in which the MAILbus 400 MTA downgrades the Common Name attribute of an O/R address exceeds the requirements defined in the standards; see Section 6.1.2.1 and Section 6.1.2.2 for details.

The MAILbus 400 MTA also downgrades the Internal Trace Information field, which is defined in the 1992 MHS Standards, so that it is compatible with that used by 1984 MOTIS implementations that conform to the CEN/CENELEC profile ENV 41201 and Chapter 7 of the OIW Stable Implementation Agreements. When the MAILbus 400 MTA receives a message containing Internal Trace Information from an MHS based on the 1984 MHS standards, it upgrades the Internal Trace Information similarly.

6.1.2 Downgrading O/R Address Attributes

The MAILbus 400 MTA implements all the downgrading procedures for removing O/R address attribute extensions as defined in Annex B of CCITT Recommendation X.419 and International Standard ISO/IEC 10021-6.

The Common Name attribute is recommended for naming users within an MTS based on the 1992 MHS Standards. However, the Common Name attribute is not present in the 1984 MHS Standards. When the MAILbus 400 MTA is communicating with systems based on the 1984 MHS Standards, in addition to mapping the Common Name attribute to a Domain Defined attribute (DDA) (as defined in the 1992 MHS Standards), it maps the Common Name to a Personal Name. The following sections describe the mapping procedures in detail and explain how they work together.

6.1.2.1 Mapping Between Common Name and Personal Name

Users in an MHS based on the 1992 MHS Standards are normally identified by a mnemonic O/R address that contains a Common Name attribute. This is a single O/R address field that contains any name by which the user is known; for example, William Davies or Bill Davies.

Users in an MHS based on the 1984 X.400 MHS are normally identified by a mnemonic O/R address that contains a Personal Name attribute. This attribute contains the following four fields:

- Surname
- Given name
- Initials
- Generation

The differences between Common Name and Personal Name attributes mean that it may not be possible to exchange messages between an MHS based on the 1992 MHS Standards and an MHS based on the 1984 X.400 MHS, that is, unless you provide appropriate information in the directory to allow the MAILbus 400 MTA to map between Common Name and Personal Name attributes (see Section 6.4 for details of the information you need to provide).

When the MAILbus 400 MTA transfers a message destined for either a peer MTA or an Agent based on the 1984 standards, and the message has a recipient or originator O/R address that contains a Common Name attribute but no Personal Name attribute, the MAILbus 400 MTA looks up the O/R address in the directory to see if there is a corresponding Personal Name registered. If a corresponding Personal Name is registered, the MAILbus 400 MTA adds it to the O/R address on the message. The MAILbus 400 MTA then deletes the Common Name attribute, and, if possible, places the Common Name value in a DDA (see Section 6.1.2.2).

6.1.2.2 Mapping Between a Common Name and a DDA

When mapping a Common Name attribute to a DDA, the MAILbus 400 MTA follows the procedure defined in the CEN/CENELEC profile ENV 41214 and the 1992 MHS Standards. This procedure preserves the Common Name attribute value of an O/R address in a DDA of type “common”. A message with a recipient O/R address that has been modified in this way can pass through a 1984 MTS and reach an MTA based on the 1992 MHS Standards, where the Common Name value will be restored to the Common Name attribute.

For this procedure to work successfully, the following conditions must apply:

- All MTAs based on the 1992 MHS Standards in the MTS make use of it.
- All 1984 MTAs and User Agents involved in handling the message are able to process O/R addresses containing DDA attributes.

- There is a DDA available in the O/R address to use for this purpose.
- The characters used in the Common Name attribute value all belong to the printable string character set and can therefore be placed in a DDA attribute.

6.1.2.3 How the Mappings for Common Name Work During Routing

This example describes how a MAILbus 400 MTA routes a message to an MTA based on the 1984 MHS Standards and maps the Common Name attribute. It also describes how any replies to the message are routed back to the MAILbus 400 MTA:

> A user in your routing domain uses the Personal Name attribute to address a recipient in a 1984 X.400 MHS. When the MAILbus 400 MTA receives a message destined for a 1984 X.400 MHS user, it normally uses the partial O/R address formed as a result of ignoring the Personal Name attribute to route the message. This is because you do not normally register all of the 1984 MHS users in the directory.
>
> On this message, the O/R address of the originator in your routing domain is likely to contain a Common Name attribute, but no Personal Name. The MAILbus 400 MTA searches the directory for a Personal Name that corresponds to the originator's Common Name attribute, and adds the Personal Name to the message. The MAILbus 400 MTA also replaces the Common Name attribute with the "common" DDA, so that the recipients can reply to the message.
>
> When the MAILbus 400 MTA receives a reply to the message from a user in the 1984 X.400 MHS, the O/R address of the recipient (that is, the user within your routing domain) has both a Personal Name attribute and a "common" DDA attribute. The MAILbus 400 MTA copies the "common" DDA value to the Common Name attribute, and deletes this DDA attribute. The MAILbus 400 MTA then uses the Common Name to find the recipient's routing information in the directory. This saves time, because the MAILbus 400 MTA does not have to search the directory to find a match for the Personal Name on the message. However, if the "common" DDA attribute is missing from the O/R address, the MAILbus 400 MTA searches the directory for a match for the Personal Name.

This example describes the action of the MAILbus 400 MTA when receiving a message sent from a user in an MTS based on the 1984 MHS standards:

> Users in a 1984 X.400 MHS use the Personal Name attribute to address a recipient in your routing domain. Alternatively, if they can enter DDA attributes, they can use the "common" DDA attribute to address the recipient. When the MAILbus 400 MTA receives a message addressed to

the recipient in your routing domain, it routes the message in the same way as for a reply to a user in your routing domain, using either the Personal Name or the “common” DDA attribute.

6.1.3 Downgrading IPMS Message Content

The 1992 MHS Standards do not define downgrading procedures for the IPMS message content.

However, the 1992 IPMS content type may not be acceptable to some 1984 messaging systems. The MAILbus 400 MTA therefore implements some downgrading procedures, which are not standardized, for IPMS message contents.

Note that these procedures are aligned with those specified in Annex C of the International Standard Profile ISO/IEC ISP 12062-1 and RFC 1328 *X.400 1988 to 1984 Downgrading*.

The MAILbus 400 MTA does not execute these procedures if any of the following conditions apply:

- The recipient O/R address is registered in the directory as receiving IPMS 1992 content type.
- The recipient O/R address is registered in the directory as receiving “any” content type.
- The originator has set the Implicit Conversion Prohibited flag on the message.

If, for any reason, the downgrade procedures fail, the MAILbus 400 MTA proceeds as if it had not invoked them, and transfers the 1992 IPMS content without modification. However, if downgrade fails at the MAILbus 400 MTA that delivers or exports the message, the MAILbus 400 MTA does not deliver the message, because the IPMS 1992 content is registered as unacceptable to the user. In this case, the MAILbus 400 MTA attempts to send a non-delivery report to the originator.

6.1.3.1 Downgrading an IPM

The MAILbus 400 MTA downgrades a 1992 IPM as follows:

1. Downgrades all O/R addresses in the IPM Heading, in the same way as for the message envelope (see Section 6.1.2.1 and Section 6.1.2.2).
2. Removes all extension fields from the IPM Heading.

3. Where possible, converts all Externally Defined bodyparts to bodyparts that are compatible with 1984 MHS Standards (see Section 6.1.3.3).
4. Unpacks all forwarded message bodyparts and applies the downgrading procedures to any nested IPM contents and delivery envelopes.

6.1.3.2 Downgrading an Interpersonal Notification

The MAILbus 400 MTA downgrades a 1992 interpersonal notification as follows:

1. Downgrades all O/R addresses in the Common Name field, in the same way as for the message envelope (see Section 6.1.2.1).
2. Downgrades all object identifiers in the Original and Converted EITs fields (according to the standardized procedure used for message envelopes).
3. Unpacks any Returned IPM field and applies the downgrading procedures described in Section 6.1.3.1.

6.1.3.3 Downgrading Externally Defined Bodyparts

The MAILbus 400 MTA converts Externally Defined bodyparts as follows:

1. Converts Externally Defined bodyparts with corresponding Basic bodypart definitions defined in the CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7 into their basic encoding. (Bodyparts that the 1992 MHS Standards refer to as “Basic” are those defined by 1984 CCITT Recommendation X.420.)
2. Converts Externally Defined bodyparts containing ODIF format to the ODA bodypart defined in the CEN/CENELEC profile ENV 41510 and Chapter 7 of the OIW Stable Implementation Agreements, December 1993.
3. Converts all remaining Externally Defined bodyparts (including General Text bodyparts) in one of the following ways:
   - Either by encapsulating Externally Defined bodyparts, in a HP-registered instance (bodypart number 62) of the USA Nationally Defined bodypart defined in Chapter 7 of the OIW Stable Implementation Agreements, December 1993.

     Note that Annex C of the International Standards Profile ISO/IEC ISP 12062-1 suggests that remaining Externally Defined bodyparts are encapsulated in basic bodyparts, such as one of the Nationally Defined bodyparts or a Bilaterally Defined bodypart. However, the MAILbus 400 MTA uses a HP-registered bodypart instead. This avoids collision with the use of other bodypart types, such as the Nationally Defined bodypart, within the 1984 MTS. This also means that the MTA can preserve Externally Defined bodyparts when they are transferred across an MTS based on the 1984 MHS standards, specifically in cases where these MTAs do not support the Interpersonal messaging 1992 content type.

From the USA Nationally Defined bodypart (number 62) the MAILbus 400 MTA can then regenerate the Externally Defined bodypart to deliver to those recipients using User Agents based on the 1992 MHS Standards.

Note

This assumes both the originator and recipient are using a User Agent based on the 1992 MHS Standards and both User Agents are connecting to a MAILbus 400 MTA. The MAILbus 400 MTA and the HP MAILbus 400 SMTP Gateway products are the only products that recognize the HP-registered instance of USA Nationally Defined bodypart (number 62).

– Or by converting Externally Defined bodyparts to Bilaterally Defined bodyparts.

  The MTA does this only if the recipient O/R address entry has the correct EIT information specified in the Content Information, see Chapter 11.

6.2 When the MTA Downgrades the Message Envelope

The MAILbus 400 MTA downgrades the message envelope when one of the following occurs:

- The Relayer is about to send a message to a peer MTA in another routing domain that uses the application context MTS Transfer Protocol 1984.

  The application context that the peer MTA uses is specified in the Peer MTA entity that holds information about the peer MTA.
- The Interface Region is about to deliver or export a message to an Agent based on the 1984 MHS standards that uses the XAPI interface.
- The Interface Region is about to export a message to an Agent that uses the Shared File 1984 interface.

6.3 When the MTA Downgrades the Message Content

The MAILbus 400 MTA downgrades the message content when the Message Processor detects that the message is for a recipient requiring 1984 IPMS content type and the conditions in Section 6.1.3 do not apply. The Message Processor performs MPDU splitting appropriately for the recipients of the message according to their content type requirements, as described in Section 3.10.

6.4 How to Set Up Your MTS to Support Downgrading

The following sections describe what you must do to allow your MTS to interwork with a peer MTA, or Agent, that conforms to the 1984 MHS Standards.

6.4.1 Registering Personal Names

Users in your routing domain can exchange mail with users in a 1984 MHS provided that all users in your routing domain have a Personal Name registered in the directory. You can register a Personal Name for a user as components of the user's Common Name entry in the directory. Use the following command to specify a Personal Name for user Bill Davies:

```
NCL> SET MTS "/MTS=ACME" -
_NCL> ORADDRESS "C=NZ;A=0;P=ACME;O=ACME;OU1=AUCK;CN=Bill Davies" -
_NCL> PERSONAL NAME "S=Davies;G=Bill;I=BHD;Q=Snr"
```

Alternatively, if you plan to use DDAs to map Common Name attribute values, and the 1984 systems have sufficient support for DDA O/R address attributes, the values that you register for the Common Name attribute for your users must contain only characters that are part of the printable string character set.

6.4.2 Setting up Boundary MTAs

In your routing domain, any boundary MTA that connects to a peer MTA in a routing domain based on 1984 MHS Standards must have a Session address. This is set up automatically as an attribute of the MTA entity if, when you run the MTA setup procedure, you answer Yes to the question that asks whether the MTA is a boundary MTA connected to 1984 MTAs.

The Peer MTA entity for a peer MTA in a 1984 routing domain must have the following:

- An application context of MTS Transfer Protocol 1984, to reflect the application context used by the peer MTA.

- A Session address that corresponds to the Session address used by the 1984 peer MTA.

6.4.3 Setting up Content Information

In order to ensure that the MTA downgrades IPMS message contents such that the message is acceptable to recipients with User Agents based on the 1984 MHS standards, you must make sure that the content information stored in the directory specifies the requirements of the recipient. See Section 11.5.1 for an example of entering content information in the directory for a recipient using a User Agent based on the 1984 MHS standards.

Part II

Tuning

This part describes how to tune a MAILbus 400 MTA. Within this guide, tuning means:

- Optimizing the way a MAILbus 400 MTA works.

  This involves modifying the characteristics of the entities in the MTA module. This is described in Chapter 7.
- Customizing how the MTA collects information about messages.

  This is described in the following chapters:
  - Chapter 8, which describes Accounting.
  - Chapter 9, which describes Archiving.
  - Chapter 10, which describes Message History logging.
- Customizing IPM bodypart conversions for your MTA or MTS.

  This is described in Chapter 11.
- Adding your own converters.

  This is described in Chapter 12.
- Customizing MTA event dispatching.

  This is described in Chapter 13.

7 Tuning an MTA

When you first run an MTA, it uses the characteristic attribute values that are either supplied with the MTA or the values you have entered in the MTA's startup script. However, after you have been running the MTAs in your routing domain for a time, you may find that you need to modify some of these attributes according to your own requirements.

Also, you may want to implement features of MTA management that do not directly relate to message throughput, for example, inserting warning text into a message received from another X.400 management domain.

This chapter explains the following:

- When to tune an MTA, see Section 7.1.
- How to monitor the message traffic in your routing domain, see Section 7.2.
- How to improve the flow of messages in your routing domain and change MPDU expiry intervals, see Section 7.3.
- How to limit the number of concurrent Peer MTA entities that are automatically created by the MTA, see Section 7.4.
- How to control the direction of message transfer between your routing domain and other routing domains, see Section 7.5.
- How to incorporate warning text into every interpersonal message (IPM) received from another X.400 management domain, see Section 7.6.
- How to control the type of network that an MTA connects to and how to use Transport characteristics other than those provided with the MTA, see Section 7.7.

Note that you can modify the MTA's characteristic attributes while the MTA is running; you do not have to disable the MTA first. These are dynamic modifications. In order to make your modifications permanent, incorporate all the appropriate commands in the MTA's startup script. For the location of this script, refer to the appendix describing the operating system specific information.

See the *MTA Module Online Help* for a description of the entities and attributes of the MTA module. The *MTA Module Online Help* also gives the values of the MTA's characteristic attributes. These are the values that are supplied with the MTA.

Note

The efficiency of an MTA depends on the capacity of the system that the MTA is running on and on the topography of your routing domain. It is therefore only possible to give general advice about modifying the characteristics of the MTA entity that improve message throughput.

7.1 Why You Need to Tune an MTA

The main reason for modifying the characteristic attributes of an MTA entity, is to enable the MTA to handle more messages simultaneously. If an MTA receives more messages than it is able to transfer, export, or deliver, then the flow of messages through the MTA slows down. When this happens the MTA becomes congested. There are several reasons why messages can be delayed inside an MTA, for example:

- An MTA reaches the limit of its outbound associations.

  An MTA can set up associations to peer MTAs until it reaches the limit specified by the Maximum Outbound Transfer Associations attribute. When an MTA is unable to create new associations, MPDUs that are awaiting transfer are delayed in the MTA.

- The MTA is having difficulty accessing the directory.

  This slows down message processing and MPDUs remain in the Processing Queue.

- An Agent or peer MTA is unavailable.

  An MPDU for that Agent or peer MTA remains in the MTA until the Agent or peer MTA is available or until the MPDU expires.

Problems relating to delays in message throughput are reported by specific events, for example, the Expiry Alarm Threshold Exceeded event. You need to investigate every occurrence of such events to find out if there is a problem with either the MTA that generated the event, the directory service, an Agent, or a peer MTA (see Section 7.2.3).

Even if a particular MTA does not generate any events relating to congestion, it might be necessary to tune one or more MTAs in your routing domain. This is because an MTA can become very busy during peak periods and occasionally reach the limits of some of its attribute values. For example, if an MTA reaches the limit specified by its Maximum Inbound Transfer Associations attribute, this temporarily prevents the MTA from accepting new associations from peer MTAs. This could cause congestion at other MTAs that have MPDUs for transfer to the MTA.

You need to identify potential congestion in your routing domain and tune one or more MTAs to prevent serious delays in the flow of messages. To identify potential congestion, you need to monitor the message traffic in your routing domain (see Section 7.2).

When you have information about the level of message traffic in your routing domain you can decide:

- Whether or not tuning is necessary.
- Which MTAs in your routing domain to tune.
- Which characteristic attributes of the MTA entity you need to modify.

Before modifying the characteristics of an MTA so that it can handle more messages concurrently, you need to be aware of:

- How changes to one attribute can affect each stage of message processing.

  If you modify one attribute so that the flow of messages from one part of the MTA is increased, then more messages are available for the next stage of message processing within the MTA. The relationship between individual attributes and each stage of message processing is explained in Section 7.3.

- How tuning one MTA is likely to affect the other MTAs in the same routing domain.

  Increasing one MTA's ability to transfer out more messages concurrently is only effective if the other MTAs in the same routing domain are able to transfer in more messages concurrently. To find out what effect tuning a single MTA has on the other MTAs in the routing domain, continue to monitor the message traffic at all the MTAs in your routing domain after you have tuned an MTA.

7.2 Monitoring Message Traffic

You can monitor the level of message traffic in your routing domain to detect any congestion by:

- Monitoring the counters that record message traffic at each MTA in your routing domain (Section 7.2.1).
- Monitoring MPDUs and the length of queues within an MTA (Section 7.2.2).
- Monitoring event sinks for specific events that indicate an MTA is congested (Section 7.2.3).

7.2.1 Using Counters to Monitor Message Traffic

You can determine the level of traffic in your routing domain by monitoring specific counters of each MTA, Agent, and Peer MTA entity in your routing domain. You monitor a counter by recording the increase in its value at frequent intervals; for example every 30 minutes, over the period of a working day. It may be necessary to repeat your observations over several days to obtain a better understanding of the flow and level of message traffic in your routing domain. To find out how many messages an MTA handles, monitor the counters listed in either Table 7–1 or in Table 7–2.

Table 7–1 Counters that Record the MPDUs an MTA Receives

Counter	Entity	Description
Submitted MPDUs	MTA	The number of messages submitted to the MTA by its User Agents and Message Stores.
Imported MPDUs	MTA	The number of messages imported by the MTA from its Gateways.
MPDUs In	Agent	The number of messages sent to the MTA by a particular Agent.
MPDUs In	Peer MTA	The number of messages the MTA has transferred in from a particular peer MTA.
Octets In	Peer MTA	The number of octets (bytes) contained in the MPDUs that the MTA has transferred in from a particular peer MTA.

Table 7–2 Counters that Record the MPDUs an MTA Sends

Counter	Entity	Description
Delivered MPDUs	MTA	The number of messages the MTA has delivered to its User Agents and Message Stores.
Exported MPDUs	MTA	The number of messages the MTA has exported to its Gateways.
MPDUs Out	Agent	The number of messages sent by the MTA to a particular Agent.
MPDUs Out	Peer MTA	The number of messages that the MTA has transferred to a particular peer MTA.
Octets Out	Peer MTA	The number of octets (bytes) contained in the MPDUs that the MTA has transferred to a particular peer MTA.

The important counters to monitor are the ones that, between them, count all the MPDUs that the MTA either sends or receives. The following commands display all the MPDUs that an MTA has received. The first command displays the counters that record all the MPDUs an MTA has received from its Agents. The second command displays the counters that record all the MPDUs an MTA has received from peer MTAs.

```
SHOW NODE "node-id" MTA SUBMITTED MPDUs, IMPORTED MPDUs
```

```
SHOW NODE "node-id" MTA PEER MTA * MPDUS IN, CREATION TIME
```

It is recommended that you display the Creation Time attribute when you display the counters of a Peer MTA entity that represents a peer MTA in your routing domain. This attribute specifies the time when the entity was created and when the counters were initialized. It is important to know when the MTA created a Peer MTA entity. This is because an MTA can delete a Peer MTA entity that it created and then create a new one with the same name. See Section 7.4 for more information about why and how an MTA creates and deletes Peer MTA entities. Note that when a Peer MTA entity is deleted, the MTA generates the Entity Deleted event. This event contains the final values of the deleted Peer MTA entity's counters. You will find this information useful to supplement the counters that you are displaying.

From the information that you obtain by monitoring the counters you can determine:

- How many messages enter and leave your routing domain over the period of your observations.
- Which are the busiest MTAs in your routing domain.
- When the peak periods occur.

If your observations reveal that an MTA is very busy, and potentially could become congested with an increase in message traffic, then you can tune it so that it can handle more messages simultaneously, see Section 7.3.

7.2.2 Monitoring MPDUs in an MTA

If you suspect that an MTA is becoming congested you can use the MPDU entity to find out about the MPDUs that are in the MTA. For example, if you receive several Expiry Alarm Threshold Exceeded events from a particular MTA, then MPDUs could be remaining in the MTA longer than they should. By finding out about the MPDUs in the MTA you might be able to identify the problem that is causing the MPDUs to be delayed.

Use the following command to display information about all the MPDUs in an MTA:

```
SHOW NODE "node-id" MTA MPDU * ALL ATTRIBUTES
```

You can also use the MPDU entity to find out about the MPDUs in a particular part of the MTA. The information supplied by an MPDU entity includes the location of the MPDU in the MTA. The location of the MPDU is provided by the State attribute of the MPDU entity.

Use the following command to find out how many MPDUs are in a particular part of the MTA:

```
SHOW NODE "node-id" MTA MPDU *, WITH STATE = value
```

where *value* is one of:

```
AWAITING PROCESSING
BEING PROCESSED
AWAITING PROCESSING RETRY
AWAITING TRANSFER
BEING TRANSFERRED
AWAITING TRANSFER RETRY
AWAITING DELIVERY OR EXPORT
BEING DELIVERED OR EXPORTED
AWAITING DELIVERY OR EXPORT RETRY
```

You can prevent MPDUs being delayed in the MTA by improving the flow of messages through the MTA. You can do this by increasing the value of one or more attributes of the MTA entity and one attribute of the Peer MTA entity. Table 7–3 lists the MPDU states that refer to parts of the MTA and the characteristic attributes that affect the flow of MPDUs through each part of the MTA.

Table 7–3 MPDU States and MTA Entity Attributes

MPDU State	Location in the MTA	Entity Attribute
Awaiting Processing	Processing Queue	Maximum Message Processors[1] (see Section 7.3.2).
Being Processed	Message Processor	Not applicable.
Awaiting Transfer	Relay Queue	Maximum Transfer Associations[1] (see Section 7.3.1.2), Maximum Outbound Transfer Associations[1] (see Section 7.3.1.5), Maximum Outbound Parallel Transfer Associations[2] (see Sections 7.3.1.6 and 7.3.1.7), Maximum Transfer Lookahead[1] (see Section 7.3.4).
Awaiting Delivery or Export	Delivery Queue	Maximum Agent Connections[1] (see Section 7.3.1.1).

[1]Attribute of the MTA entity

[2]Attribute of the MTA entity and the Peer MTA entity

7.2.3 Events That Indicate Congestion in an MTA

Some events, if they occur frequently, can indicate that an MTA is working to the limits specified by one or more of its characteristic attributes. For example, an MTA can accept the maximum number of inbound associations allowed by the value of its Maximum Inbound Transfer Associations attribute. When an MTA reaches the limit specified by this attribute, the MTA rejects association requests from peer MTAs. Consequently, peer MTAs that have MPDUs for that MTA can become congested because they are unable to transfer MPDUs to that MTA. If an MPDU is unable to leave an MTA, it eventually expires.

The following events indicate that MPDUs are being delayed in an MTA or peer MTA:

- Inbound Transfer Soft Rejection (see also Section 16.2.2)

  Different error messages contained in this event indicate which limit the MTA has reached, as follows:

  - The error message `Maximum Associations Reached` indicates that the MTA has reached the limit specified by the Maximum Transfer Associations attribute of the MTA entity. The MTA is temporarily unable to accept new inbound associations.
  - The error message `Maximum Inbound Transfer Associations Reached` indicates that the MTA has reached the limit specified by the Maximum Inbound Transfer Associations attribute of the MTA entity. The MTA is temporarily unable to accept new inbound associations.
  - The error message `Maximum Inbound Parallel Transfer Associations Reached` indicates that the MTA has reached the limit specified by the Maximum Inbound Parallel Transfer Associations attribute of the Peer MTA entity that represents the peer MTA that is attempting to establish an association. This error message is only generated when the peer MTA requesting the association is in another routing domain. The MTA is temporarily unable to accept new inbound associations from the peer MTA.

  If this event occurs frequently, then increase the value of the relevant characteristic attribute at the MTA that generated the event (see Sections 7.3.1.2 to 7.3.1.4).

- Outbound Soft Rejection (see also Section 16.2.4)

  If this event has the error message `RTSE Busy`, then the peer MTA with which the MTA is trying to set up an association is congested. If the peer MTA continues to reject association requests from the MTA, then MPDUs for that peer MTA can remain in the MTA until they expire.

  If the peer MTA that is rejecting association requests is in your routing domain, then tune the peer MTA and increase the value of its Maximum Inbound Transfer Associations attribute, see Section 7.3.1.3.

If the peer MTA that is rejecting association requests is in another routing domain, then inform the person responsible for managing the peer MTA. Possible ways to prevent association requests being rejected by the peer MTA could involve:

  - Tuning the peer MTA in the other routing domain so that it can accept more concurrent associations from the boundary MTA.
  - Increasing the number of MTAs in the other routing domain that the boundary MTA can communicate with.

- Rejected Agent Connection (see also Section 19.3.1)

  If this event has the error message `Maximum Connections Exceeded`, then the MTA has reached the limit specified by its Maximum Agent Connections attribute. When the number of Agent connections reaches the limit specified by this attribute, Agents are unable to connect to the MTA until an existing connection from an Agent is released. Any MPDUs that the MTA has for delivery or export to Agents that are unable to make connections remain in the MTA.

  If this event occurs frequently, then tune the MTA by increasing the value of its Maximum Agent Connections attribute, see Section 7.3.1.1.

- Expiry Alarm Threshold Exceeded (see also Section 18.7.1)

  This is a warning that an MPDU in the MTA is about to expire. This event provides the following information:

  - The MPDU's target; that is, the next destination of the MPDU.
  - The state of the MPDU; that is, the location of the MPDU in the MTA.

  If you receive several Expiry Alarm Threshold Exceeded events, each identifying the same target, then this could indicate that a particular peer MTA or Agent is unavailable. See if there are any other events in the event sink that might provide information about why the target identified by this event is unavailable.

  If you receive several Expiry Alarm Threshold Exceeded events each identifying different targets, then the MTA that generated the event could be congested. The information provided about the state of the MPDU corresponds to the values of the State attribute of the MPDU entity. You can use the MPDU entity to find out how many MPDUs are in that part of the MTA, see Section 7.2.2.

- MPDU Expired (see also Section 18.7.3)

  Investigate each occurrence of this event and find out what is causing an MPDU to remain in the MTA long enough for it to expire.

7.3 Improving the Flow of Messages

You can improve the flow of messages through an MTA by:

- Increasing the number of connections and associations that the MTA can have at any one time.
- Increasing the number of Message Processors available to the MTA.
- Decreasing the amount of time the MTA waits between attempts to establish an association.
- Improving the way the MTA selects MPDUs for transfer.
- Decreasing the amount of time that associations remain idle.

Before modifying the characteristics that affect an MTA's ability to handle more messages concurrently, you need to consider how your changes are going to affect the operation of the MTA as a whole. For example, if you enable more messages to enter the MTA concurrently, then the Processing Queue might become congested. To prevent this, you must also increase the number of Message Processors so that the MTA can process more messages concurrently. To do this, you might need more system resources or more directory capacity. See Section 3.1 for more information about the conditions that affect the efficiency of running Message Processors in parallel. However, enabling more Message Processors might mean that the Relay Queue becomes congested. Therefore, you may also need to enable more messages to leave the MTA concurrently.

After you make one or more changes to the attributes that determine message throughput, find out how your changes have affected the MTA. Do this by monitoring the various queues in the MTA, using the MPDU entity, as described in Section 7.2.2.

7.3.1 Increasing Connections and Associations

The number of MPDUs that can enter an MTA concurrently and leave an MTA concurrently is controlled by the following attributes:

- The Maximum Agent Connections attribute of the MTA entity (Section 7.3.1.1)
- The Maximum Transfer Associations attribute of the MTA entity (Section 7.3.1.2)
- The Maximum Inbound Transfer Associations attribute of the MTA entity (Section 7.3.1.3)

- The Maximum Inbound Parallel Transfer Associations attribute of the Peer MTA entity (Section 7.3.1.4)
- The Maximum Outbound Transfer Associations attribute of the MTA entity (Section 7.3.1.5)
- The Maximum Outbound Parallel Transfer Associations attribute of the MTA entity (Section 7.3.1.6) and of the Peer MTA entity (Section 7.3.1.7)

The MTA uses a large number of Transport connections. You are advised to set the Maximum Transport Connections attribute of the OSI Transport module, based on the sum of the following MTA entity attributes:

- Twice the value of the Maximum Transfer Associations attribute
- Three times the value of the Maximum Agent Connections attribute
- The value of the Maximum Message Processors attribute

If you modify any of these MTA entity attributes, you are advised to modify the value of the Maximum Transport Connections attribute accordingly.

Note that the value of the Maximum Remote NSAPs attribute must be greater than the value of Maximum Transport Connections, both of which are attributes within the OSI Transport module.

OpenVMS

On OpenVMS systems, you also need to set the SYSGEN parameter CHANNELCNT to at least:

(2 x Maximum Transport Connections) + 160.

♦

7.3.1.1 Maximum Agent Connections

This attribute specifies the number of User Agents or Gateways that can be connected to the MTA at any one time.

If possible, set the value of this attribute to equal the total number of registered and unregistered Agents of the MTA.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM AGENT CONNECTIONS value
```

where *value* is the maximum number of Agents that can be connected to the MTA at any one time.

7.3.1.2 Maximum Transfer Associations

This MTA entity attribute specifies the total number of concurrent associations that an MTA can have. This applies to all associations, both to and from peer MTAs, either within the routing domain or in another routing domain.

If an MTA reaches the limit set by this attribute, it cannot create or accept a new association until one of the following occurs:

- An association to a peer MTA is released or aborted.
- An idle association from a peer MTA is either released or aborted by the peer MTA that initiated it or is aborted by the MTA.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM TRANSFER ASSOCIATIONS value
```

where *value* is the maximum number of concurrent associations.

The value of this attribute overrides the values of the following attributes if, either individually or collectively, they are set higher:

- Maximum Inbound Transfer Associations

  In turn, the value of the Maximum Inbound Transfer Associations attribute overrides the value of the Maximum Inbound Parallel Transfer Associations attribute of the Peer MTA entity if the latter attribute is set higher.

- Maximum Outbound Transfer Associations

  In turn, the value of the Maximum Outbound Transfer Associations attribute overrides the value of the following attributes if they are set higher:

  - The Maximum Outbound Parallel Transfer Associations attribute of the MTA entity
  - The Maximum Outbound Parallel Transfer Associations attribute of the Peer MTA entity

The following examples show the relationship between the attributes that control the number of concurrent associations to and from an MTA:

1. The following attribute values enable the MTA to operate more efficiently than in example 2:

   Maximum Transfer Associations = 20
   Maximum Inbound Transfer Associations = 10
   Maximum Outbound Transfer Associations = 10
   Maximum Outbound Parallel Transfer Associations = 5

Using these attribute values, the MTA can accept and create associations up to the number that you specify for each individual attribute. For example, at any one time, the MTA can have ten inbound associations and ten outbound associations. Note that the Maximum Outbound Parallel Transfer Associations attribute value is a subset of the Maximum Outbound Transfer Associations attribute value. This means that concurrently the MTA could have associations to a maximum of ten different peer MTAs, one association to each, or to a minimum of two peer MTAs, five associations to each.

2. An MTA that is set up with the following attribute values could create delays in the transfer of messages:

   Maximum Transfer Associations = 15
   Maximum Inbound Transfer Associations = 25
   Maximum Outbound Transfer Associations = 25
   Maximum Outbound Parallel Transfer Associations = 25

   Note that the maximum value of all attributes is 15 not 25, this is because the value of the Maximum Transfer Associations attribute overrides the other attributes. Also, the following problems could occur:

   – The MTA could accept 15 concurrent inbound associations and be unable to create any outbound associations.
   – The MTA could create 15 concurrent outbound associations and be unable to accept any inbound associations.
   – The MTA could set up 15 concurrent outbound associations to the same peer MTA and be unable to accept or create any associations to or from other peer MTAs.

7.3.1.3 Maximum Inbound Transfer Associations

This MTA entity attribute specifies the maximum number of concurrent associations that an MTA can have from any number of peer MTAs either within the routing domain or in another routing domain. This affects the number of messages that the MTA can transfer in concurrently.

If an MTA reaches the limit set by this attribute, it cannot accept a new association from a peer MTA until an idle inbound association is either released by the peer MTA that initiated it or is aborted by the MTA.

It is important to set similar values for this attribute and the Maximum Outbound Transfer Associations attribute. If you set a high value for this attribute and a very low value for the Maximum Outbound Transfer Associations attribute, the MTA is likely to become congested. The effect is to cause the MTA to transfer in more messages than it can transfer out.

Always set the value of this attribute to be *higher* than the value of the Maximum Inbound Parallel Transfer Associations attribute of any Peer MTA entity.

See Section 7.3.1.2 for examples of the relationship between this attribute and the other attributes that specify the maximum number of associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM INBOUND TRANSFER ASSOCIATIONS value
```

where `value` is the maximum number of concurrent inbound transfer associations.

7.3.1.4 Maximum Inbound Parallel Transfer Associations

This Peer MTA entity attribute specifies the maximum number of concurrent associations that a boundary MTA can have from a specific peer MTA in another routing domain. This affects the number of messages that the boundary MTA can transfer in concurrently from the specified peer MTA.

For each peer MTA in a routing domain, this attribute and its outbound equivalent (the Maximum Outbound Parallel Transfer Associations attribute of the Peer MTA entity, described in Section 7.3.1.7) limit the flow of traffic between the boundary MTA and the specific peer MTA. These limits have most significance on the boundary between X.400 management domains, where the cost of MPDU transfer is important. See Section 7.3.1.6 for details of the attribute that limits the flow of traffic between your MTA and any single peer MTA within your routing domain.

If the number of associations to a boundary MTA from a particular peer MTA in another routing domain reaches the limit set by this attribute, the MTA cannot accept any more associations from the peer MTA.

If possible, set the value of this attribute to be the same as, or higher than, the outbound association capacity of the peer MTA. This avoids unnecessary failures when the Peer MTA is establishing associations.

Set the value of this attribute to be *lower* than the value of the Maximum Inbound Transfer Associations attribute of the MTA entity. This prevents the MTA using up its quota of inbound associations for one peer MTA.

See Section 7.3.1.2 for examples of the relationship between this attribute and the other attributes that specify the maximum number of associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA PEER MTA -
[TYPE = MANUALLY CONFIGURED, NAME = "peer-mta-name"] -
MAXIMUM INBOUND PARALLEL TRANSFER ASSOCIATIONS value
```

where `peer-mta-name` is the name of the Peer MTA entity and `value` is the maximum number of concurrent inbound associations that the boundary MTA can accept from the peer MTA.

7.3.1.5 Maximum Outbound Transfer Associations

This MTA entity attribute specifies the maximum number of concurrent associations that an MTA can have to any number of peer MTAs either within the routing domain or in another routing domain. This affects the number of messages that the MTA can transfer out concurrently.

If an MTA reaches the limit set by this attribute, it cannot create a new outbound association until an association to a peer MTA becomes idle. When an association to a peer MTA becomes idle and the MTA has an MPDU awaiting transfer to this particular peer MTA, then the MTA re-uses the association. When an association to a peer MTA becomes idle and the MTA has an MPDU for transfer to another peer MTA, the MTA releases the idle association immediately in order to set up a new association to the other peer MTA.

It is important to set similar values for this attribute and the Maximum Inbound Transfer Associations attribute. If you set a low value for this attribute and a very high value for the Maximum Inbound Transfer Associations attribute, the MTA is likely to become congested. The effect is to cause the MTA to transfer in more messages than it can transfer out.

Always set the value of this attribute to be *higher* than the value of the Maximum Outbound Parallel Transfer Associations attributes (of both the MTA entity and any Peer MTA entity). This prevents the MTA using up its quota of outbound associations on transferring messages to a single peer MTA.

See Section 7.3.1.2 for examples of the relationship between this attribute and the other attributes that specify the maximum number of associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM OUTBOUND TRANSFER ASSOCIATIONS value
```

where `value` is the maximum number of concurrent outbound transfer associations.

7.3.1.6 Maximum Outbound Parallel Transfer Associations to Peer MTAs Within Your Routing Domain

This MTA entity attribute specifies the maximum number of concurrent associations that an MTA can have to a single peer MTA within the same routing domain.

This attribute limits the flow of traffic between your MTA and any single peer MTA within your routing domain. Within your routing domain, all MTAs are MAILbus 400 MTAs, so the setting for this attribute specified at the other MTAs in the routing domain (that is the peer MTAs within the routing domain) implicitly controls the number of inbound parallel transfer associations for this MTA. See Section 7.3.1.4 and Section 7.3.1.7 for details of the attributes that limit the flow of traffic between your MTA and specific peer MTAs in other routing domains.

If the number of associations from an MTA to a particular peer MTA within the routing domain reaches the limit set by this attribute, the MTA cannot establish another association to the peer MTA.

Set the value of this attribute to be *lower* than the value of the Maximum Outbound Transfer Associations attribute. This prevents the MTA using up its quota of outbound associations for one peer MTA.

See Section 7.3.1.2 for examples of the relationship between this attribute and the other attributes that specify the maximum number of associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM OUTBOUND PARALLEL TRANSFER ASSOCIATIONS value
```

where *value* is the maximum number of concurrent outbound associations to the same peer MTA.

7.3.1.7 Maximum Outbound Parallel Transfer Associations to Peer MTAs in Other Routing Domains

This Peer MTA entity attribute specifies the maximum number of concurrent associations that a boundary MTA can have to a single peer MTA in another routing domain. This affects the number of messages that the boundary MTA can transfer concurrently to a single peer MTA in another routing domain.

For each peer MTA in a routing domain, this attribute and its inbound equivalent (the Maximum Inbound Parallel Transfer Associations attribute of the Peer MTA entity, described in (Section 7.3.1.4)) limit the flow of traffic between the boundary MTA and the specific peer MTA. These limits have most significance on the boundary between X.400 management domains, where the cost of MPDU transfer is important. See Section 7.3.1.6 for details of the attribute that limits the flow of traffic between the boundary MTA and any single peer MTA within your routing domain.

If the number of associations from a boundary MTA to a particular peer MTA in another routing domain reaches the limit set by this attribute, the MTA cannot establish another association to the peer MTA.

Set the value of this attribute to be *lower* than the value of the Maximum Outbound Transfer Associations attribute of the MTA entity. This prevents the MTA using up its quota of outbound associations for one peer MTA.

If the peer MTA is in the same X.400 management domain as your MTA, you may want to set the value of this attribute to be the same as the value for the Maximum Outbound Parallel Transfer Associations attribute of the MTA entity. This ensures that all peer MTAs within the X.400 management domain have the same capacity for transferring messages, irrespective of the routing domain in which they are located.

See Section 7.3.1.2 for examples of the relationship between this attribute and the other attributes that specify the maximum number of associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA PEER MTA -
[TYPE = MANUALLY CONFIGURED, NAME = "peer-mta-name"] -
MAXIMUM OUTBOUND PARALLEL TRANSFER ASSOCIATIONS value
```

where *peer-mta-name* is the name of the Peer MTA entity and *value* is the maximum number of concurrent outbound associations to the peer MTA.

7.3.2 Increasing the Number of Message Processors

The number of MPDUs that an MTA can process concurrently is determined by the number of Message Processors that the MTA can have at any one time. The number of concurrent Message Processors is specified by the Maximum Message Processors attribute of the MTA entity.

Note that increasing the number of Message Processors increases the throughput of MPDUs, provided that certain conditions are met. For example, if an MTA is converting several messages, then increasing the number of Message Processors can increase the throughput of messages. This is because a Message Processor does not access the directory when converting messages. A Message Processor that is converting messages will therefore not compete for directory access with other Message Processors.

Increasing the number of Message Processors also improves the message handling capacity of an MTA that serves a large number of local users and does not transfer messages to peer MTAs or export messages to Gateways.

However, running several Message Processors could decrease message throughput if all Message Processors try to access the directory concurrently. See Section 3.1 for more information about the conditions that affect the efficiency of running Message Processors in parallel.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM MESSAGE PROCESSORS value
```

where *value* is the number of Message Processors.

7.3.3 Modifying Association Idle Intervals

An MTA can maintain associations after they become idle. An association becomes idle after a complete MPDU has been transferred over the association. The MTA maintains the association for use later on as this saves time and resources setting up a new association when there is another MPDU to transfer over the association.

The following attributes specify the length of time that associations can be maintained by an MTA after the transfer of a complete MPDU:

- Maximum Idle Outbound Transfer Association Interval (Section 7.3.3.1)
- Maximum Idle Inbound Transfer Association Interval (Section 7.3.3.2)

7.3.3.1 Maximum Idle Outbound Transfer Association Interval

This attribute specifies how long the MTA maintains an idle association to a peer MTA. The MTA releases the association before the specified time has elapsed if it needs to set up an association to a different peer MTA but has reached its quota of outbound associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM IDLE OUTBOUND TRANSFER ASSOCIATION -
INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the maximum time that the MTA can maintain the association.

Set this attribute to zero (0-00:00:00) if you want the MTA to release outbound associations as soon as an MPDU has been transferred and there are no other MPDUs queued for the peer MTA.

7.3.3.2 Maximum Idle Inbound Transfer Association Interval

This attribute specifies how long the MTA maintains an idle association from a peer MTA. Once this time has elapsed, the MTA aborts the association and generates the Inbound Failure event (see Section 16.2.6). The MTA does not abort the association before this time has elapsed. Consequently, if the number of concurrent associations to the MTA reaches the limit set by the Maximum Inbound Transfer Associations attribute, then the MTA cannot accept a new association until one of the following occurs:

- A peer MTA releases an association it has made to the MTA.
- The MTA aborts an idle inbound association when the time specified by this attribute has elapsed.

Set the value of this attribute to be higher than the time specified by the Maximum Idle Outbound Transfer Association Interval at each MTA in your routing domain. This ensures that inbound associations from peer MTAs in your routing domain are released by the initiating MTA before they can be aborted by the receiving MTA.

When setting this attribute at a boundary MTA, find out how the peer MTA in the other routing domain responds when an association it initiated becomes idle. If the peer MTA maintains the association after it becomes idle, then set this attribute to be higher than the idle interval used by the peer MTA for its outbound associations.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM IDLE INBOUND TRANSFER ASSOCIATION -
INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the time interval.

Set this attribute to zero (0-00:00:00) if you want the MTA to wait until the peer MTA releases the association.

7.3.4 Changing the Way an MTA Selects MPDUs for Transfer

An MTA selects an MPDU for transfer to a particular peer MTA according to the priority of the MPDU and how long the MPDU has been in the MTA. This is an unbiased way of selecting MPDUs, provided that the MTA can transfer the MPDU that it selects. However, an MTA is unable to transfer the most eligible MPDU when both the following occur:

- The MTA does not have an idle association to the target peer MTA.
- The MTA is unable to create a new association to the target peer MTA because it has reached the limit specified by one of the following attributes:
  - Maximum Transfer Associations (Section 7.3.1.2)
  - Maximum Outbound Transfer Associations (Section 7.3.1.5)
  - Maximum Outbound Parallel Transfer Associations (Section 7.3.1.6)

The MTA must wait until it can either create a new association to the target peer MTA or an existing association to the target peer MTA becomes idle. The MTA then re-uses this association.

To prevent delays occurring, the MTA can search for an alternative MPDU to transfer to a different peer MTA. Note that the MTA only needs to search for an alternative MPDU when it is prevented from transferring the most eligible MPDU. The extent of the MTA's search is controlled by the Maximum Transfer Lookahead attribute of the MTA entity.

7.3.4.1 Maximum Transfer Lookahead Attribute

The Maximum Transfer Lookahead attribute does not come into effect unless all of the following conditions are true:

1. An existing association becomes idle.
2. The value set for either Maximum Transfer Associations or Maximum Outbound Transfer Associations is reached.
3. There is an MPDU in the Transfer Queue awaiting transfer.

Use the Maximum Transfer Lookahead attribute to specify the maximum number of MPDUs that can be overlooked in favor of an MPDU further down the Transfer Queue whose target matches that of the idle association.

An MPDU is overlooked if the associations to the target are at their limit. The overlooked MPDU will be transferred when an existing association to the target becomes idle.

The MTA starts counting MPDUs, against the Maximum Transfer Lookahead value, from the first MPDU that is eligible for transfer, and that requires a new association, but cannot be transferred as the values set for Maximum Transfer Associations and Maximum Outbound Associations have already been reached.

The MTA looks for an MPDU whose target matches the target of the idle association. If no MPDU is found, the idle association is disconnected and a new association is set up to the target for the most eligible MPDU.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM TRANSFER LOOKAHEAD value
```

where `value` is the number of MPDUs, in addition to the MPDU at the head of the queue, that the MTA can include in its search.

Set this attribute to zero to prevent the MTA from making a search for an MPDU to transfer to a different peer MTA.

7.3.5 Modifying Retry Intervals

When an MTA is unable to establish an association to a peer MTA or when an established association to a peer MTA fails, the MTA places MPDUs for that peer MTA in a retry set. See Section 4.3.2 for more information about retry sets. The MPDUs remain in the retry set until one of the following occurs:

- The MTA establishes or recovers the association and transfers the MPDUs to the peer MTA.
- The MPDUs expire.

How frequently the MTA makes a retry attempt to establish an association or to recover an association to the peer MTA depends on the value of the retry interval. The retry interval is the time that the MTA waits before making the first and any subsequent retry attempts. The retry interval has a minimum and maximum value that you can set. The retry interval is increased each time a retry attempt fails, until the maximum value is reached. The minimum and maximum values of the retry interval are specified by the following attributes of the MTA entity:

- Initial Transfer Retry Interval

  This attribute specifies the length of time that the MTA waits before making the first retry attempt.

  Use the following command to specify this interval:

  ```
  SET NODE "node-id" MTA INITIAL TRANSFER RETRY INTERVAL d-hh:mm:ss
  ```

  where *d-hh:mm:ss* is the time interval.

- Maximum Transfer Retry Interval

  This attribute specifies the maximum length of time that the MTA can wait between retry attempts.

  Use the following command to specify this interval:

  ```
  SET NODE "node-id" MTA MAXIMUM TRANSFER RETRY INTERVAL d-hh:mm:ss
  ```

  where *d-hh:mm:ss* is the time interval.

The MTA doubles the retry interval after each failed retry attempt until one of the following occurs:

- The retry interval reaches the value specified by the Maximum Transfer Retry Interval attribute.
- If the retry interval were doubled, it would exceed the value specified by the Maximum Transfer Retry Interval attribute.

When one of these conditions occurs, the retry interval remains at the value specified by the Maximum Transfer Retry Interval attribute.

For example, an MTA is set up with the following values:

Initial Transfer Retry Interval = 5 minutes
Maximum Transfer Retry Interval = 30 minutes

When an association fails and the MTA makes several attempts to recover that association, then the retry interval varies according to Table 7–4:

Table 7–4 Example of How the MTA Calculates the Retry Interval

Retry Attempt	Retry Interval (minutes)	Result of Retry Attempt	Comment
1	5	Fails	The MTA waits until the time specified by the Initial Transfer Retry Interval attribute has elapsed before trying to recover the association. Because the first retry attempt fails, the MTA doubles the retry interval to 10 minutes.
2	10	Fails	Because the second retry attempt fails, the MTA doubles the retry interval to 20 minutes.

(continued on next page)

Table 7–4 (Cont.) Example of How the MTA Calculates the Retry Interval

Retry Attempt	Retry Interval (minutes)	Result of Retry Attempt	Comment
3	20	Fails	Because the third retry attempt fails, the MTA tries to double the retry interval. However, if the MTA were to double the retry interval it would exceed the value specified by the Maximum Transfer Retry Interval attribute. Therefore, the MTA sets the retry interval to 30 minutes. This is the value specified by the Maximum Transfer Retry Interval attribute.
4	30	Fails	Because the fourth retry attempt fails and the Maximum Retry Interval has already been reached, the retry interval remains at 30 minutes.
5	30	Succeeds	The MTA starts to transfer the first MPDU in the retry set. If the transfer is successful, then the MTA sets the retry interval to 5 minutes. This is the value specified by the Initial Transfer Retry Interval. If the transfer of the first MPDU in the retry set fails, then the MTA does not change the retry interval but makes another retry attempt to the peer MTA after a further retry interval of 30 minutes has elapsed.

The values that you should use for the Initial Transfer Retry Interval or the Maximum Transfer Retry Interval attributes depend on the type of network that the MTA is part of. If the MTA is part of a wide area network (WAN), then use values of several minutes. If the MTA is part of a local area network (LAN), then you can use values of less than a minute.

You can monitor retry attempts to a particular peer MTA using the following attributes of the Peer MTA entity that represents the peer MTA:

- Retry Count

  This attribute records the number of retry attempts made by the MTA to establish or recover an association to the peer MTA. When a retry attempt to the peer MTA is successful, the MTA resets this attribute to zero.

- Retry Time

  This attribute holds the time of the next retry attempt. If there are no MPDUs in the Retry Queue for the peer MTA, then this attribute has the same value as the Creation Time attribute.

Use the following command to display these attributes:

```
SHOW NODE "node-id" MTA PEER MTA identifier RETRY COUNT, RETRY TIME
```

where *identifier* is either:

- `[TYPE = MANUALLY CONFIGURED, NAME = "`*peer-mta-name*`"]`

  where *peer-mta-name* is the name of the Peer MTA entity.

- `[TYPE = AUTOMATICALLY CONFIGURED, NAME = "`*name*`"]`

  where *name* is the name of the peer MTA's entry in the directory.

7.3.6 Changing MPDU Expiry Intervals

You can specify a limit for the local processing of an MPDU. This time is calculated from when the complete MPDU enters the MTA. The MPDU expires if it is in the MTA longer than the time you specify. When the MPDU expires, the MTA attempts to send a non-delivery report to the originator of the MPDU. See Section 1.2.1 for details of how the MTA uses MPDU expiry times.

To specify how long an MPDU can remain in the MTA, modify the Local MPDU Expiry Interval attribute of the MTA entity, as described in Section 7.3.6.1.

To ensure that message transfer is consistent throughout your routing domain, apply the same Local MPDU Expiry Interval at all MTAs in your routing domain.

You can also specify how long an MPDU is allowed to take to reach its destination. The time is calculated from when the MPDU enters the first ADMD on its route. The length of time can be varied for MPDUs that have different priorities. The following attributes of the MTA entity specify the appropriate expiry interval:

- Nonurgent MPDU Expiry Interval (Section 7.3.6.2)
- Normal MPDU Expiry Interval (Section 7.3.6.3)
- Urgent MPDU Expiry Interval (Section 7.3.6.4)

Setting CCITT Expiry Intervals

The MTA's startup script contains NCL commands that set these attributes to the expiry times recommended by the CCITT. To use the commands in the startup script remove the "!" from the start of each command.

For the location of this script, refer to the appendix describing the operating system specific information.

7.3.6.1 Local MPDU Expiry Interval

This attribute specifies how long an MPDU can remain in the MTA. Use the following command to modify this attribute:

```
SET NODE "node-id" MTA LOCAL MPDU EXPIRY INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the time interval.

The time that you specify for the Local MPDU Expiry Interval attribute determines the expiry alarm threshold. The MTA automatically sets this threshold to be half the value of the Local MPDU Expiry Interval attribute. The MTA generates the Expiry Alarm Threshold Exceeded event when an MPDU remains in the MTA longer than this threshold. The Expiry Alarm Threshold Exceeded event provides a warning that an MPDU is about to expire.

7.3.6.2 Nonurgent MPDU Expiry Interval

This attribute specifies how long MPDUs that have a nonurgent priority can take to reach their destinations.

The MTA's startup script contains an NCL command that sets this attribute to the expiry time recommended by the CCITT. If you want to use the command, then edit the MTA's startup script, as described in the note in Section 7.3.6.

Otherwise, use the following command to set this attribute:

```
SET NODE "node-id" MTA NONURGENT MPDU EXPIRY INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the expiry interval.

7.3.6.3 Normal MPDU Expiry Interval

This attribute specifies how long MPDUs that have a normal priority can take to reach their destinations.

The MTA's startup script contains an NCL command that sets this attribute to the expiry time recommended by the CCITT. If you want to use this command, then edit the MTA's startup script, as described in the note in Section 7.3.6.

Otherwise, use the following command to set this attribute:

```
SET NODE "node-id" MTA NORMAL MPDU EXPIRY INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the expiry interval.

7.3.6.4 Urgent MPDU Expiry Interval

This attribute specifies how long MPDUs that have an urgent priority can take to reach their destinations.

The MTA's startup script contains an NCL command that sets this attribute to the expiry time recommended by the CCITT. If you want to use this command, then edit the MTA's startup script, as described in the note in Section 7.3.6.

Otherwise, use the following command to set this attribute:

```
SET NODE "node-id" MTA URGENT MPDU EXPIRY INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the expiry interval.

7.4 Controlling the Number of Peer MTA Entities

The Maximum Automatically Configured Peer MTAs attribute of the MTA entity limits the number of Peer MTA entities that the MTA can have and that are created by the MTA. This attribute has no effect on the number of concurrent Peer MTA entities manually created to represent peer MTAs in other routing domains.

An MTA creates a Peer MTA entity automatically when it first communicates with a particular peer MTA in the same routing domain. The MTA creates a Peer MTA entity to hold information about the messages that it exchanges with the peer MTA. The maximum number of Peer MTA entities that the MTA needs to create is related to the number of MTAs in the routing domain. For example, if a routing domain contains ten MTAs, then each MTA could create up to nine Peer MTA entities.

It is not possible to manually delete Peer MTA entities that are automatically created by the MTA. You can control the number of concurrent Peer MTA entities created by the MTA by specifying the maximum number that can exist concurrently.

The maximum number of concurrent Peer MTA entities that are automatically created by the MTA is specified by the Maximum Automatically Configured Peer MTAs attribute. When the number of Peer MTA entities, created by the MTA, reaches the limit specified by this attribute, the MTA deletes the oldest Peer MTA entity that has no subordinate Activity entities. See Section 1.3.3 for more information about how the MTA applies the limit specified by this attribute.

When an MTA deletes a Peer MTA entity, it generates the Entity Deleted event. This event contains the final values of the counters for the deleted Peer MTA entity.

Note

The value of the Maximum Automatically Configured Peer MTAs attribute does not affect potential or existing associations with peer MTAs.

Use the following command to modify this attribute:

```
SET NODE "node-id" MTA MAXIMUM AUTOMATICALLY CONFIGURED PEER MTA value
```

where `value` is the maximum number of concurrent Peer MTA entities automatically created by the MTA.

7.5 Controlling Message Transfer to Other Routing Domains

At a boundary MTA, you can control the direction of message transfer to and from another routing domain that the boundary MTA is connected to. The direction of message transfer is specified by the Direction attribute of the Peer MTA entity that holds information about a peer MTA in the other routing domain. This attribute has one of the following values:

- Inbound

  The boundary MTA can receive messages from the peer MTA in the other routing domain but cannot send messages to this peer MTA.

- Outbound

  The boundary MTA can send messages to the peer MTA in the other routing domain but cannot receive messages from this peer MTA.

- Inbound and Outbound

  The boundary MTA can send messages to and receive messages from the peer MTA in the other routing domain.

Use the following command to specify the direction of transfer to and from a peer MTA in another routing domain:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] DIRECTION value
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain and *value* is the direction of transfer.

7.6 Inserting Warning Text into Messages Received from Other X.400 Management Domains

The MTA can insert text, written by you, into each interpersonal message (IPM) that it receives from routing domains in other X.400 management domains. Your text appears at the start of the IPM. Use your warning text to notify your users that a particular message they have received is from another organization, for example:

```
WARNING: This message has originated outside of your
   organization. Keep this in mind when replying.
```

Note that inserting your text into an IPM is an optional feature that has to be manually set up. If you tune an MTA to insert your text, it inserts it into all the IPMs it receives from all other X.400 management domains.

The MTA identifies a routing domain in another X.400 management domain from the information about that routing domain held in the directory. The Different CCITT Domain attribute of the Domain entity indicates whether or not a particular routing domain is part of another X.400 management domain. This attribute must be set to “TRUE” in order for the MTA to recognize that the routing domain is in another X.400 management domain.

7.6.1 Creating Your Warning Text

Your warning text is held in the MTA’s warning text file. This is one of the files that is installed with the MTA. Refer to the appendix describing the operating system specific information for the name and location of this file.

OpenVMS

After you have set up the MTA, `MTA$IA5_WARNING_TEXT.TXT` is moved from `SYS$COMMON` to the device and directory used by the MTA, and specified when the MTA is set up: *device:*`MTA$`*node*.
♦

After installation, the MTA's warning text file does not contain any text; this means that no warning text is inserted by the MTA. If you want the MTA to automatically insert your warning text into an IPM do the following:

1. Edit the file so that it contains your warning text.

   Your warning text can be of any length, but must contain only IA5 characters to ensure that it can be received by all users. Also note that the MTA inserts the complete contents of the file. It is not possible to comment out any of the text in the file.

2. Stop and restart the MTA.

   See the appropriate appendix for information about how to stop and start the MTA.

If you want to stop the MTA inserting your warning text into messages, either remove the text or delete the MTA's warning text file, then stop and start the MTA.

Note

If you delete the MTA's warning text file, and, in the future, you want the MTA to insert your warning text into an IPM, recreate the file so that it has the same file specification as the warning text file installed with the MTA. See the appropriate appendix for the location of the MTA's warning text file.

OpenVMS

Note that if you create a new warning text file you should ensure that the filename and location are those given in Table E–1 and that the new file has the RMS record format Stream_LF.
♦

7.7 Customizing the MTA's Use of the Transport Service

There are two types of Transport Service that the MTA can use:

- DECnet/OSI Transport Service
- TCP/IP Transport Service

The TCP/IP Transport Service is provided by use of the RFC 1006 protocol, which emulates the OSI Transport Service Class 0 over a TCP/IP network. The RFC 1006 protocol enables OSI applications, such as the MTA, to use the TCP/IP network.

7.7.1 How MTAs Use the Transport Service

The address of an MTA can be either a Presentation or Session address. A Presentation or Session address can contain several network service access points (NSAPs). Each NSAP contains a network address that is associated with a particular type of network service. An MTA connects to a peer MTA using the appropriate network address provided by an NSAP in the peer MTA's address.

Section 7.7.1.1 describes the DECnet/OSI Transport Service connections. Section 7.7.1.2 describes the TCP/IP Transport Service connections.

7.7.1.1 The DECnet/OSI Transport Service

There are two network services available for DECnet/OSI Transport connections:

- Connectionless Network Service (CLNS), used for a local area network (LAN)

  For local area network connections to a peer MTA, the peer MTA's Presentation or Session address must contain a CLNS NSAP.

- Connection-oriented Network Service (CONS), used for a wide area network (WAN)

  For wide area network connections to a peer MTA, the peer MTA's Presentation or Session address must contain a CONS NSAP.

Note that a peer MTA's address can contain both a CLNS and CONS NSAP. The MTA selects the NSAP to use according to the value of the Network Service attribute supplied in a Transport Template entity. For example, if the Transport Template entity specified by the MTA defines the network service as CLNS, the Transport Service uses a CLNS NSAP in the peer MTA's Presentation or Session address. For more information about the MTA's use of Transport Templates see Section 7.7.4.

7.7.1.2 The TCP/IP Transport Service

For TCP/IP Transport Service connections to a peer MTA, the peer MTA's Presentation or Session address must contain an RFC 1006 NSAP.

The TCP port number defined in RFC 1006 to listen for RFC 1006 connections is 102. RFC 1006 NSAPs that are generated when the MTA is set up contain 102 as defined in RFC 1006.

As an example, if you have the TCP/IP address 1.2.3.4 the NSAP is:

```
RFC1006+1.2.3.4+102,RFC1006
```

The following are examples of how an RFC 1006 NSAP can be expressed:

RFC1006+1.2.3.4+102
RFC1006+1.2.3.4,RFC1006
RFC1006+1.2.3.4+102,RFC1006

These RFC 1006 NSAPs are equivalent. You can omit the port number, in which case the TCP port number 102 is assumed, and "RFC1006" at the end of the NSAP is optional. When displayed using NCL the form of the RFC1006 NSAP is always:

```
NS+54007287220318800001103400102000001,RFC1006"
```

Refer to the DECnet-Plus documentation set for the rules for encoding RFC1006 NSAPs.

7.7.2 How an MTA Selects a Transport Service

Tru64 UNIX

On Tru64 UNIX, the MTA entity contains a Transport Service Options attribute that specifies the Transport Service that an MTA can use for outbound connections to peer MTAs in the same routing domain.

A Peer MTA entity also contains a Transport Service Options attribute. Use this attribute to specify the Transport Service to be used by a boundary MTA for connections to a peer MTA in another routing domain. If the Transport Service Options attribute of a Peer MTA entity is set to null, the boundary MTA uses the Transport Service specified by its MTA entity.

The Transport Service Options attribute can contain the values OSI and TCPIP. The value OSI refers to the DECnet/OSI Transport Service; the value TCPIP refers to the TCP/IP Transport Service. When you set up the MTA, the Transport Service Options attribute of the MTA entity is automatically set up.

The order of the values in the Transport Service Options attribute is important. For example, if the attribute specifies OSI before TCPIP, the MTA always tries to use the DECnet/OSI Transport Service first.

For information about changing the order of the Transport Services specified by the MTA's Transport Service Options attribute, and how to set the Transport Service Options attribute of a Peer MTA entity, see Section 7.7.3.

♦

OpenVMS

On OpenVMS, you can define Transport Template entities for each type of DECnet/OSI network on a particular node including an RFC1006 network type; see Section 7.7.4.

On OpenVMS the MTA does not use the Transport Service Options attribute. If you want to specify a preference in the order that the MTA uses each Transport Service, you must modify either the MTA or the peer MTA attribute “Template Name” and include the individual Transport Templates in the order that you want the connection attempts tried; see Section 7.7.5.

♦

7.7.3 Modifying the Transport Service Options Attributes

Tru64 UNIX

This section explains how to modify the Transport Service Options attribute of the MTA entity and the Peer MTA entity on Tru64 UNIX.

You can modify the Transport Service Options attribute to either change the order of the Transport Services it specifies or to specify only one Transport Service.

Use the following command to set the Transport Service Options attribute of an MTA entity:

```
SET NODE "node-id" MTA TRANSPORT SERVICE OPTIONS -
(value[, value])
```

where *value* is one of TCPIP or OSI.

Do not set this attribute to null; if you do, the MTA is unable to make outbound connections to peer MTAs in the same routing domain.

For connections to a peer MTA in another routing domain, use the Transport Service Options attribute of a Peer MTA entity to specify a different Transport Service or order of Transport Services than that specified by the MTA entity.

Use the following command to set the Transport Service Options attribute of a Peer MTA entity:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY -
CONFIGURED, NAME = "peer-mta-name"] TRANSPORT SERVICE -
OPTIONS (value[, value])
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain and *value* is one of TCPIP or OSI.

Note

Any changes you make to the Transport Service Options attribute in the MTA or Peer MTA entities must also be included in the MTA's startup script.

OpenVMS

On OpenVMS, as the MTA does not use the Transport Service Options attribute, if you try to use the following NCL commands, NCL returns an Invalid Attribute error:

- SET MTA TRANSPORT SERVICE OPTIONS
- SET PEER MTA TRANSPORT SERVICE OPTIONS
- SHOW MTA TRANSPORT SERVICE OPTIONS
- SHOW PEER MTA TRANSPORT SERVICE OPTIONS

♦

7.7.4 MTA OSI Transport Templates

Within DECnet/OSI Transport management, the characteristics of OSI Transport communication over a particular type of network service are defined by a Transport Template entity. Transport Template entities are set up on each node that is connected to the network as entities of the OSI Transport module. See the DECnet/OSI network management documentation for more information about the OSI Transport module.

An individual node can be connected to more than one type of network service, and there must be at least one Transport Template entity set up for each type of DECnet/OSI network on the node. For example, if a node is connected to a local area network and a wide area network there is:

- A Transport Template entity with the Network Service attribute set to CLNS.

  This Transport Template entity describes the Transport characteristics for a local area network.

- A Transport Template entity with the Network Service attribute set to CONS.

This Transport Template entity describes the Transport characteristics for a wide area network.

- A Transport Template entity with the Network Service attribute set to RFC1006 (OpenVMS).

When a DECnet/OSI application, in this case an MTA, makes use of the DECnet/OSI Transport Service to set up a Transport connection to a peer MTA located at another node, it can specify the name of a Transport Template entity. This Transport Template entity defines the characteristics of the OSI Transport connection. One of the characteristics of a connection is the Transport Class (TP0, TP2, or TP4) to be used. In the case of the MTA, these Transport Template entities are specific to the MTA and are created when an MTA is installed and set up. The MTA then uses its own Transport Template entities in preference to the Transport Template entities defined by DECnet/OSI.

7.7.4.1 Inbound Communication

For inbound connections using the DECnet/OSI Transport Service, the MTA's use of a Transport Template depends on the operating system that the MTA is running on.

Tru64 UNIX

When the MTA is installed on a Tru64 UNIX system, a script is installed and executed that defines the Transport Template entity "mta_any" for all inbound communications. The "mta_any" Transport Template entity supports Transport protocol classes 0 and 4 for a CONS network connection and Transport protocol class 4 for a CLNS network connection. The "mta_any" Template entity has no associated attribute in the MTA entity or subentities, so you cannot change the name of this template.

♦

OpenVMS

On OpenVMS systems, the MTA does not use a Transport Template entity for inbound communication but can accept Transport protocol classes 0, 2, and 4 for inbound CONS and CLNS network connections.

♦

7.7.4.2 Outbound Communication

For outbound connections using the DECnet/OSI Transport Service, a script is executed that defines one or both of the following Transport Template entities within the MTA's routing domain:

- "mta_clns" for outbound communications across a LAN
- "mta_cons" for outbound communications across a WAN
- "mta_rfc1006" for outbound communications over TCP/IP (OpenVMS)

You can modify the attributes of all the MTA's Transport Template entities with the exception of the Network Service attribute. As an example, you might want to change the Transport Class defined in the Transport Template. See the DECnet/OSI documentation for information about how to modify these attributes.

Note

Any changes you make to the OSI Transport Templates automatically created by the MTA must be included in the MTA's Transport Template scripts. For the name and location of these scripts, refer to the appendix describing the operating system specific information.

Section 7.7.4.3 describes the use of Transport Templates within the routing domain and Section 7.7.4.4 describes the use of Transport Templates at boundary MTAs.

7.7.4.3 Within the Routing Domain

The names of the Transport Template entities used for outbound communication within the routing domain are defined in the Template Name attribute of the MTA entity. This is done when the MTA startup file is executed. Note that the "mta_cons" Transport Template name is included in the Template Name attribute even if the corresponding Transport Template entity is not created. This does not cause a problem if your network service is CLNS only. The sequence of Transport Template entity names in the Template Name attribute is significant; the first Transport Template entity specified by the MTA is the one that the Transport Service attempts to use first.

There are circumstances where you might want to change the sequence in which the templates appear in the Template Name attribute. For example, if the MTAs in your routing domain are connected to a CONS network, and you never want your MTAs to use the CLNS network, you can remove the "mta_clns" Transport Template entity name specified in the Template Name attribute at each MTA in your routing domain. Section 7.7.5 describes the commands that you use to modify the Template Name attribute at each MTA.

7.7.4.4 At Boundary MTAs

You can also specify the names of the Transport Template entities that a boundary MTA uses for outbound communication with a peer MTA in another routing domain. You specify these in the Template Name attribute of the Peer MTA entity that represents the peer MTA. If you do not specify the name of a Transport Template entity in the Template Name attribute of the Peer MTA entity, the boundary MTA uses the Transport Template entities specified in the Template Name attribute of it's MTA entity.

Be aware that if you modify a boundary MTA's use of Transport Template entities in any way, and there are no Transport Template entities specified in Peer MTA entities, you might find that a boundary MTA can no longer make connections to a peer MTA. For example, the MTAs in your routing domain are using CLNS. You have decided to remove the "mta_cons" Transport Template entity name from each MTA's Template Name attribute and at the boundary MTA there is no Template Name attribute set up for the Peer MTA entities. This boundary MTA can therefore only make connections to peer MTAs in the other routing domains using CLNS. In this case, in order to use CONS to a particular MTA, you must include the "mta_cons" Transport Template entity name in the Template Name attribute of the Peer MTA entity that represents the peer MTA in the other routing domain. This is then used in preference to the Template Name provided for the boundary MTA. Alternatively, you can name another suitable Transport Template entity with the Network Service attribute set to CONS.

7.7.5 Modifying Template Name Attributes

The Template Name attribute of the MTA entity specifies the DECnet/OSI network services that can be used by the MTA within the local routing domain. Note that this attribute is automatically set up to contain the values "mta_clns", "mta_cons" and "mta_rfc1006" (OpenVMS). Modify this attribute if you want the MTA to use a different sequence of network services or another Transport Template entity that exists on the node. Use the following command to modify this attribute:

```
SET NODE "node-id" MTA TEMPLATE NAME ("template-name", "template-name")
```

where *template-name* is the name of a Transport Template entity. Note that this attribute can specify any number of OSI Transport Templates.

If, for any reason, the MTA entity's Template Name attribute does not contain a value, outbound communication is affected as follows:

Tru64 UNIX

On Tru64 UNIX systems, the Template Name attribute can be null. If no value is supplied for the Template Name attribute, the MTA uses the DECnet/OSI network service specified by the Transport Template entity called “DEFAULT” for outbound communication.

♦

OpenVMS

On OpenVMS systems, the Template Name attribute must contain the name of a Transport Template entity for DECnet/OSI connections, otherwise the MTA is unable to make outbound connections over DECnet/OSI.

♦

The type of DECnet/OSI network service to be used by the MTA to connect to a particular peer MTA in another routing domain is specified by the Template Name attribute of the Peer MTA entity. Note that this attribute is not automatically set up. Use the following command to modify the Template Name attribute of the Peer MTA entity:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] TEMPLATE NAME ("template-name", "template-name")
```

where:

- *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain.
- *template-name* is the name of an OSI Transport Template entity.

This attribute can specify any number of OSI Transport Templates.

If the Peer MTA entity’s Template Name attribute does not contain a value, the MTA uses the network services specified by the Template Name attribute of the MTA entity.

Note

Any changes you make to the Template Name attributes of the MTA and Peer MTA entities must be included in the MTA’s startup script.

8 Accounting

This chapter describes how you can use Accounting in your routing domain.

Accounting is a feature that is used to record certain items of information about the messages exchanged between an MTA and its Agents and also between the MTA and peer MTAs in other X.400 routing domains.

Note

No Accounting information is logged about messages transferred between the MTAs within a routing domain. You can obtain information about such message traffic by monitoring the following counters: MPDUs In, MPDUs Out, Octets In, and Octets Out.

Section 7.2 describes how to do this in more detail.

To record Accounting information you need to enable Accounting. Enabling Accounting activates the Accounting filters, which are characteristic attributes of the MTA entity and its relevant Peer MTA entities. The Accounting filters are listed in Table 8–1. They determine the specific items of information that are recorded from the messages that enter and leave an MTA. If required, this information can be used as a basis for generating message traffic statistics.

An MTA provides automatic management for Accounting in that it purges Accounting information at regular intervals. However, you should be aware that the process of writing Accounting information can reduce an MTA's performance. Accounting files also reduce the disk space available on the system where the MTA is running.

Section 8.1 gives examples of why you might want to collect Accounting information and where in your routing domain you would enable Accounting. Section 8.2 describes how to use Accounting in your routing domain.

8.1 Examples of Using Accounting

Use the examples in this section as a guide for assessing whether and where to collect Accounting information at your routing domain boundaries.

8.1.1 Charging a Third Party for Message Transfer Services

An organization providing a message transfer service for third parties can use Accounting information as a basis for invoicing its customers. Such invoices could be based on Accounting information reflecting the amount of message traffic, and the size and priority of each message. In this case, Accounting is enabled at the MTAs which handle messages for the third party. These could be the MTAs that receive third party messages, the MTAs that pass these on to the target routing domain, or both.

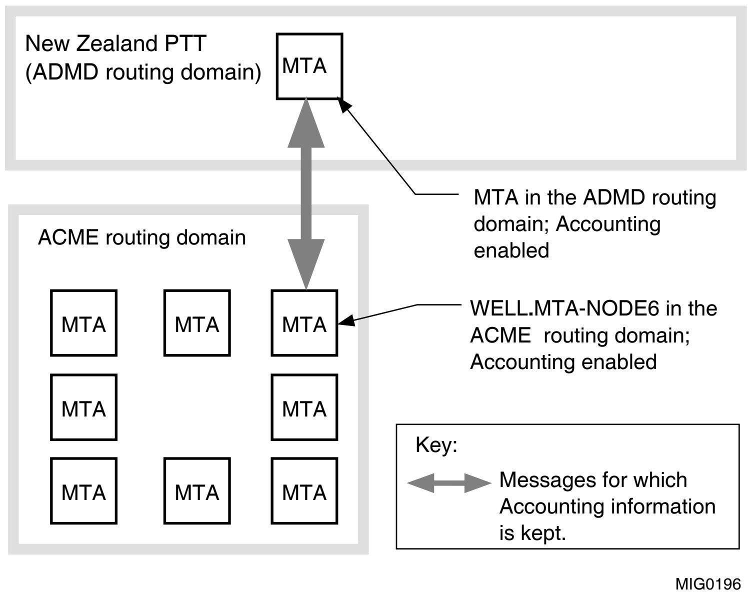
The following example uses the ACME Shoe Corporation, a fictitious shoe manufacturer in New Zealand. The message transfer between the ACME Shoe Corporation and other organizations is handled by the ADMD New Zealand PTT. To collect the information required for invoicing ACME, the New Zealand PTT has enabled Accounting at the MTA that connects to the ACME routing domain. Figure 8–1 illustrates the New Zealand PTT's use of Accounting with regard to ACME.

8.1.2 Controlling Charges for Message Transfer Services

An organization using the service of a third party for message exchange with other organizations (for example ACME Shoe Corporation as described in Section 8.1.1) can use Accounting as a means of ensuring that it is invoiced correctly. In this case, Accounting needs to be enabled at each MTA that connects to the third party providing the message transfer service.

ACME Shoe Corporation uses Accounting for this purpose. See Figure 8–1.

Figure 8–1 Accounting Used for the ADMD Connection


New Zealand PTT
(ADMD routing domain)
MTA
MTA in the ADMD routing domain; Accounting enabled
ACME routing domain
MTA
MTA
MTA
WELL.MTA-NODE6 in the ACME routing domain; Accounting enabled
MTA
MTA
MTA
MTA
MTA
Key:
Messages for which Accounting information is kept.
MIG0196

8.1.3 Cross-Charging Between Departments

An organization can use Accounting to record internal message traffic, for example, if one department in the organization is responsible for handling inter-departmental message transfer for the entire routing domain. Accounting information identifies the originator of a message, and therefore the cost for the transfer of a message can be charged to the originator's department. In this case, Accounting information would be collected at all MTAs that are connected to User Agents.

8.1.4 Creating Statistics About the Workload of an MTA

You can use Accounting to obtain statistical information about the amount of message exchange between an MTA and its Agents, and peer MTAs in other routing domains. For example, an organization may have set up an MTS based on the MAILbus 400 MTA and wants to establish whether it is operating properly and is laid out efficiently. In this case, Accounting could be enabled temporarily at all relevant MTAs in the routing domain. Evaluating the resulting data can help to plan modifications to the routing domain.

You can also use an MTA's counter attributes to obtain information about the volume of internal message traffic (see Section 7.2). However, Accounting information contains more details about the individual messages that are processed by your MTAs (for example message size).

8.2 How to Tune Accounting

You can tune Accounting as follows:

- Enable or disable Accounting at an MTA (Section 8.2.1).
- Choose the information that is logged by modifying Accounting filter settings (Section 8.2.2).
- Change the interval at which Accounting information is purged (Section 8.2.3).
- Process Accounting files (Section 8.2.4).

To ensure that Accounting information is reliable and consistent, change Accounting settings as part of a policy that spans your entire routing domain. This means making identical changes to the Accounting settings of all MTAs where Accounting is enabled, unless you have specific reasons to change Accounting settings at a particular MTA only.

8.2.1 Enabling or Disabling Accounting

When you enable Accounting at the MTA entity, each of the filters listed in Table 8–1 is activated and logs information, if a corresponding connection exists. When you disable Accounting, these filters are deactivated.

Use the following command to enable or disable Accounting at an MTA:

```
SET NODE "node-id" MTA ACCOUNTING STATE value
```

where *value* is either `ON` (enable) or `OFF` (disable).

You cannot switch individual Accounting filters off, but you can set these filters so that no information is being logged (see Section 8.2.2).

Table 8–1 MTA Accounting Filters

Filter	Description
Submission Accounting Filter[1]	Determines the information logged at message submission from a User Agent.
Delivery Accounting Filter[1]	Determines the information logged at message delivery to a User Agent.
Import Accounting Filter[1]	Determines the information logged at message import from a Gateway.
Export Accounting Filter[1]	Determines the information logged at message export to a Gateway.
Transfer In Accounting Filter[2]	Determines the information logged when a message is transferred in from a particular peer MTA in another routing domain.
Transfer Out Accounting Filter[2]	Determines the information logged when a message is transferred out to a particular peer MTA in another routing domain.

[1]Characteristic attribute of the MTA entity

[2]Characteristic attribute of the Peer MTA entity

Accounting filters have settings that determine the items of information that are logged from a message. These are listed in Table 8–2 and Table 8–3.

8.2.2 Choosing Accounting Filter Settings

To modify the information that is logged by an Accounting filter, add or remove Accounting filter settings, as required. The Accounting filter settings are listed in Table 8–2 and Table 8–3.

Use the ADD and REMOVE commands as described in the *MTA Module Online Help* to add or remove Accounting filter settings, or use the SET command to replace existing settings.

For example, use the following command to remove filter settings from the Export Accounting filter:

```
REMOVE NODE "node-id" MTA EXPORT ACCOUNTING FILTER -
       {setting, setting}
```

where *setting* is an Accounting filter setting.

To enable a particular Accounting filter to log all possible items of information, include all unused filter settings in the ADD command. To prevent a particular Accounting filter from logging any information, include all used filter settings in the REMOVE command.

Table 8–2 and Table 8–3 list all Accounting filter settings and show:

- The filter settings that log information about messages, probes and reports
- The filter settings that are used in a particular Accounting filter
- The filter settings that are the defaults supplied with the MTA

Use these tables to decide which Accounting filter settings to add or remove. Some filter settings log information from message fields on the X.400 message envelope (as defined in CCITT Recommendation X.411 and in International Standard ISO/IEC 10021-4). Other filter settings log information related particularly to the MTA where Accounting is used.

Table 8–2 Information Logged about Messages and Probes

Settings Relating to the MAILbus 400 MTA	
Filter Setting	**Applicable Filter**
Message Size[1]	All
Agent [1]	Submission, Delivery, Import and Export Accounting Filter
Domain [1]	Transfer In and Transfer Out Accounting Filter

Settings Corresponding to Message Fields Defined in X.411 and ISO/IEC 10021-4	
Filter Setting	**Applicable Filter**
Message Originator [1]	All
Message DL[2] Expansion History	All except Submission Accounting Filter
Message Recipient Information [1]	All
Message Recipient Redirection History	All except Submission Accounting Filter
Message Flags	All
Message External Trace Information	All
Message Internal Trace Information	All
Message Priority [1]	All
Message Content Type	All

[1]Default supplied with the MTA

[2]DL = Distribution list

(continued on next page)

Table 8–2 (Cont.) Information Logged about Messages and Probes

Settings Corresponding to Message Fields Defined in X.411 and ISO/IEC 10021-4	
Filter Setting	**Applicable Filter**
Message EITs	All

Table 8–3 Information Logged about Reports

Settings Relating to the MAILbus 400 MTA	
Filter Setting	**Applicable Filter**
Report Size [1]	All
Agent [1]	Submission, Delivery, Import and Export Accounting Filter
Domain [1]	Transfer In and Transfer Out Accounting Filter
Settings Corresponding to Message Fields Defined in X.411 and ISO/IEC 10021-4	
Filter Setting	**Applicable Filter**
Report Subject Identifier	All
Reported Originator and DL[2] Expansion History [1]	All except Submission Accounting Filter
Report Destination Name [1]	All
Reported Actual and Intended Recipient[1]	All
Report External Trace Information	All
Report Internal Trace Information	All

[1]Default supplied with the MTA

[2]DL = Distribution list

Some of the Accounting filter settings contain timestamps. These are listed in Table 8–4. Use these filter settings if you want to use timestamps when processing Accounting files (see Section 8.2.4 for information about processing Accounting files).

Table 8–4 Timestamps in Accounting Filter Settings

Filter Setting	Timestamp
Message DL[1] Expansion History	Records time when expansion occurred
Message Recipient Redirection History	Records time when redirection occurred
Message External Trace Information	Records time of arrival at X.400 management domain
Message Internal Trace Information	Records time of arrival at MTA
Report External Trace Information	Records time of arrival at X.400 management domain
Report Internal Trace Information	Records time of arrival at MTA

[1]DL = Distribution list

8.2.3 Changing the Accounting Purge Interval

An MTA creates new Accounting files as required. The Accounting Purge Interval attribute automatically purges Accounting files that exceed a certain age. Unless modified, the value of this attribute is seven days, which means that Accounting files older than seven days are deleted.

For Accounting files to be purged, the MTA must be enabled and the purge interval must not be zero (0-00:00:00). When the Accounting Purge Interval is zero, purging is disabled.

Before you change the Accounting Purge Interval, you need to find out how much disk space Accounting files occupy at an MTA in a day. You then need to check this figure against the availability of disk space at that MTA. Set the Accounting Purge Interval to purge Accounting files so that there is always sufficient disk space available.

If you are already using Accounting, monitor the size of Accounting files in the Accounting directory. For the location of the Accounting directory, see the appendix describing the operating system specific information.

If you have not used Accounting, you can use the following method to determine the disk space that may be required for your Accounting files:

Estimate the average size of Accounting records in Accounting files (in bytes) and the average number of messages that the relevant MTAs handle in a day. If your MTS is already operational, you can use the following counters to help you determine the number of messages:

- Counters of the MTA entity:

  Delivered MPDUs
  Exported MPDUs
  Imported MPDUs
  Submitted MPDUs

- Counters of Peer MTA entities that are manually configured:

  MPDUs In
  MPDUs Out

To determine the average Accounting record size, decode an Accounting file, using the Accounting Decoder tool, and count the number of records in it. Divide the total size of the Accounting file by the number of records within it to obtain an average Accounting record size. Multiplying the number of messages with the average Accounting record size gives you an indication of the amount of disk space that Accounting files can occupy in a day. Refer to the appendix describing the operating system specific information for information about how to use the Accounting Decoder tool.

When you decide to change the Accounting Purge Interval, make identical changes at all MTAs where Accounting is enabled, unless you have specific reasons to change the setting at a particular MTA only. When modifying the Accounting Purge Interval, ensure that you do not cause the MTA to delete Accounting information before it is used.

Use the following command to modify the Accounting Purge Interval:

```
SET NODE "node-id" MTA ACCOUNTING PURGE INTERVAL d-hh:mm:ss
```

where `d-hh:mm:ss` is the desired purge interval in binary relative time. Only the value you enter for `d` is used; that is, the value you supply for the number of days.

8.2.4 Processing Accounting Files

Accounting information is stored in ASN.1 Basic Encoding Rules (BER) format in Accounting files. ASN.1 is a standard form of notation used to represent data and is described in CCITT Recommendations X.208 and X.209 as well as International Standards ISO 8824 and ISO 8825 (see Appendix A). Section 8.3 describes the ASN.1 BER in Accounting files.

The Accounting Decoder tool enables you to translate the ASN.1 BER data in Accounting files to text.

You can build a utility that processes Accounting information, either in ASN.1 format or in translated text format, to compile invoices and statistics reflecting the number of messages processed by your routing domain. See Section 8.3 if you intend to write a utility that reads the ASN.1 BER data stored in Accounting files.

Accounting files are kept in a specific directory. For the location of the Accounting directory, refer to the appendix describing the operating system specific information. The unique name of an Accounting file is ensured by mapping the current date to the file name.

The following are examples of Accounting file names:

Tru64 UNIX

```
1994032101
```

♦

OpenVMS

```
1994032101.
```

Note that the file name is terminated by a period.

♦

In the above examples `1994` is the year, `03` is the month, and `21` is the day when the Accounting file was created. The last two digits (`01` in the above examples) are incremented throughout the day as new Accounting files are created.

An MTA requires disk space to store Accounting information. You may therefore consider copying Accounting files to another storage medium regularly for post-processing. This enables you to have Accounting files purged immediately after they have been copied. Alternatively, you could process Accounting information while it is still on an MTA's disk before it is purged.

If you receive the System Interface Error event in combination with other events indicating data losses, and if counters indicating such data losses increase suddenly and substantially, investigate whether this is due to Accounting files occupying too much disk space. If this is the case Accounting files must be purged or moved to another storage medium more frequently.

8.3 Accounting File Format

Accounting information in an Accounting file is in ASN.1 (BER) format. To use the information in this section you must already be familiar with handling encoded ASN.1 data.

The information that is written to Accounting files is determined by Accounting filter settings, such as Message Originator or Message Content Type. These Accounting filter settings cause information to be logged from message attributes as defined in CCITT Recommendation X.411 and International Standard ISO 10021-4. In addition, the Accounting filter settings log information that is specific to the MTA where Accounting is used.

Table 8–5 lists the message attributes that are defined in X.411 and ISO/IEC 10021-4 and that are written to Accounting files for messages or probes.

Table 8–5 Accounting Filter Settings in Messages and Probes and Corresponding X.411 and ISO/IEC 100021-4 Message Attributes

Accounting Filter Setting	X.411 and ISO/IEC 10021-4 Message Attribute
No particular setting required[1]	message-identifier, probe-identifier
Message Originator	originator-name
Message DL[2] Expansion History	dl-expansion-history
Message Recipient Redirection History	redirection-history
Message Flags	per-message-indicators
	recipient-reassignment-prohibited
	dl-expansion-prohibited
	conversion-with-loss-prohibited
Message External Trace Information	trace-information
Message Internal Trace Information[3]	internal-trace-information
Message Priority[4]	priority
Message Content Type	content-type

[1]The message-identifier and probe-identifier attributes are always logged when Accounting is enabled

[2]DL = Distribution list

[3]ISO DIS 8883 MOTIS 84 Internal Trace elements are not recorded in the Trace Information

[4]For messages only

(continued on next page)

Table 8–5 (Cont.) Accounting Filter Settings in Messages and Probes and Corresponding X.411 and ISO/IEC 100021-4 Message Attributes

Accounting Filter Setting	X.411 and ISO/IEC 10021-4 Message Attribute
Message EITs	original-encoded-information-types or trace-information
Message Recipient Information	recipient-name
	per-recipient-indicators
	requested-delivery-method
	physical-forwarding-prohibited[4]
	physical-forwarding-address-request[4]
	physical-delivery-modes[4]
	registered-mail-type[4]
	physical-delivery-report-request[4]
	proof-of-delivery-request[4]

[4]For messages only

Table 8–6 lists the message attributes that are defined in X.411 and ISO/IEC 10021-4 and that are written to Accounting files for reports.

Table 8–6 Accounting Filter Settings in Reports and Corresponding X.411 and ISO/IEC 10021-4 Message Attributes

Accounting Filter Setting	X.411 and ISO/IEC 10021-4 Message Attribute
No particular setting required	report-identifier[1]
Report Subject Identifier	subject-identifier
Reported Originator and DL[2] Expansion History	originator-and-dl-expansion-history
Report Destination Name	report-destination-name

[1]The report-identifier attribute is always logged when Accounting is enabled

[2]DL = Distribution list

(continued on next page)

Table 8–6 (Cont.) Accounting Filter Settings in Reports and Corresponding X.411 and ISO/IEC 10021-4 Message Attributes

Accounting Filter Setting	X.411 and ISO/IEC 10021-4 Message Attribute
Reported Actual and Intended Recipient	actual-recipient-name and originally-intended-recipient-name
Report External Trace Information	trace-information
Report Internal Trace Information[3]	internal-trace-information

[3]ISO DIS 8883 MOTIS 84 Internal Trace elements are not recorded in the Trace Information

8.3.1 ASN.1 in Accounting Files

This section does not provide a complete specification of the ASN.1 for the MTA accounting files. Definitions not included in this section can be found in the X.400 standard X.411 and ISO/IEC 10021-4.

The following ASN.1 describes the format of Accounting files as used by the MTA.

```
--   ASN.1 taken from Recommendation X.419 and ISO/IEC 10021-6
--   MTS application protocol data units

MTS-APDU ::= CHOICE {
                message             [0] Message,
                probe               [2] Probe,
                report              [1] Report }


--   ASN.1 taken from Recommendation X.411 and ISO/IEC 10021-4 and annotated
--   for use in Accounting files

Message ::= SEQUENCE {
            envelope MessageTransferEnvelope }
Probe   ::= ProbeTransferEnvelope
Report  ::= SEQUENCE {
            envelope ReportTransferEnvelope,
            content ReportTransferContent }

--   Message transfer envelope

MessageTransferEnvelope ::= SET {
     COMPONENTS OF PerMessageTransferFields,
     per-recipient-fields           [2] SEQUENCE SIZE (1..ub-recipients) OF
                    PerRecipientMessageTransferFields }


PerMessageTransferFields ::= SET {
     accounting-information              AccountingInformation,
     --   not taken from Recommendation X.411;
     --   Accounting information for the MTA
     message-identifier                  MessageIdentifier,
     originator-name                     OriginatorName OPTIONAL,
     original-encoded-information-types  OriginalEncodedInformationTypes
                                          OPTIONAL,
     content-type                        ContentType OPTIONAL,
     priority                            Priority DEFAULT normal,
     per-message-indicators              PerMessageIndicators DEFAULT {},
     trace-information                   TraceInformation OPTIONAL,
     extensions                       [3] EXTENSIONS CHOSEN FROM {
          recipient-reassignment-prohibited,
          dl-expansion-prohibited,
          conversion-with-loss-prohibited,
          dl-expansion-history,
          internal-trace-information } DEFAULT {} }

PerRecipientMessageTransferFields ::= SET {
     recipient-name                       RecipientName,
     per-recipient-indicators         [1] PerRecipientIndicators,
     extensions                       [3] EXTENSIONS CHOSEN FROM {
          requested-delivery-method,
          physical-forwarding-prohibited,
          physical-forwarding-address-request,
          physical-delivery-modes,
          registered-mail-type,
          physical-delivery-report-request,
          proof-of-delivery-request } DEFAULT {} }

--   Probe transfer envelope

ProbeTransferEnvelope ::= SET {
     COMPONENTS OF PerProbeTransferFields,
     per-recipient-fields           [2] SEQUENCE SIZE (1..ub-recipients) OF
                                        PerRecipientProbeTransferFields }

PerProbeTransferFields ::= SET {
     accounting-information               AccountingInformation,
     --   not taken from Recommendation X.411;
     --   Accounting information for the MTA

     probe-identifier                     ProbeIdentifier,
     originator-name                      OriginatorName OPTIONAL,
     original-encoded-information-types   OriginalEncodedInformationTypes
                                           OPTIONAL,
     content-type                         ContentType OPTIONAL,
     per-message-indicators               PerMessageIndicators DEFAULT {},
     trace-information                    TraceInformation OPTIONAL,
     extensions                       [3] EXTENSIONS CHOSEN FROM {
          recipient-reassignment-prohibited,
          dl-expansion-prohibited,
          conversion-with-loss-prohibited,
          dl-expansion-history,
          internal-trace-information } DEFAULT {} }
```

```
PerRecipientProbeTransferFields ::= SET {
     recipient-name                         RecipientName,
     per-recipient-indicators          [1] PerRecipientIndicators,
     extensions                        [3] EXTENSIONS CHOSEN FROM {
          requested-delivery-method } DEFAULT {} }

--   Report transfer envelope

ReportTransferEnvelope ::= SET {
     accounting-information                  AccountingInformation,
     --   not taken from Recommendation X.411;
     --   Accounting information for the MTA

     report-identifier                       ReportIdentifer,
     report-destination-name                 ReportDestinationName OPTIONAL,
     trace-information                       TraceInformation OPTIONAL,
     extensions                        [1] EXTENSIONS CHOSEN FROM {
          originator-and-DL-expansion-history,
          internal-trace-information } DEFAULT {} }

--   Report transfer content

ReportTransferContent ::= SET {
     COMPONENTS OF PerReportTransferFields,
     per-recipient-fields           [0] SEQUENCE SIZE (1..ub-recipients) OF
                                        PerRecipientReportTransferFields }

PerReportTransferFields ::= SET {
     subject-identifier                      SubjectIdentifier OPTIONAL }

PerRecipientReportTransferFields ::= SET {
     actual-recipient-name               [0] ActualRecipientName,
     originally-intended-recipient-name  [4] OriginallyIntendedRecipientName
                                              OPTIONAL }
```

Accounting Information Particular to the MTA

```
AccountingInformation ::= [PRIVATE 0] IMPLICIT AccountingSequence

     AccountingSequence ::= SEQUENCE {
   1 accounting-time               INTEGER,
   2 accounting-point              AccountingPoint,
   3 content-size              [0] IMPLICIT INTEGER OPTIONAL,
   4 message-source            [1] IMPLICIT MsgSource OPTIONAL,

   5 CHOICE {
            domain-name                [2] IMPLICIT PrintableString,
            registered-agent-name      [3] IMPLICIT PrintableString,
            mta-or-set-name            [4] IMPLICIT PrintableString,
            unregistered-agent-name    [5] IMPLICIT ORName
            } OPTIONAL
            }
```

```
AccountingPoint ::= ENUMERATED {
         submission        (0),
         delivery          (1),
         import            (2),
         export            (3),
         transfer-in       (4),
         transfer-out      (5)
         }

MsgSource ::= SET {
         type              [0] IMPLICIT MsgSourceType,
         string            [1] IMPLICIT IA5String
         }

MsgSourceType ::= ENUMERATED {
         domain            (1),
         registered-agent  (2),
         MTA               (3),
         unregistered-agent (5)
         }
```

where

1 `accounting-time` is time in seconds since 00:00:00 Greenwich Mean Time (GMT), January 1, 1970.

2 `accounting-point` indicates where data has been logged during message processing. It can take any of the values defined in the `AccountingPoint` enumeration.

3 `content-size` is logged only if the Accounting filter settings Message Size or Report Size are used. The content size is in Kilobytes.

4 `message-source` indicates from where a message entered the MTA. The message source is indicated by:

- `type`, which can take any of the values defined in the `MsgSourceType` enumeration
- `string`, which is the name of the domain, MTA, registered Agent or unregistered Agent indicated in `type`.

Note that message source information is only logged for instances where the AccountingPoint is either submission (0), import (2), or transfer in (4), and that no destination information is logged in these instances.

5 The elements of the `CHOICE` indicate the destination of a message.

Note that destination information is only logged for instances where the AccountingPoint is either delivery (1), export (3), or transfer out (5), and that no message source information is logged in these instances.

For definitions of any elements not defined in this section refer to CCITT Recommendation X.411.

9 Archiving

This chapter describes how you can use Archiving in your routing domain. It also describes the format of an archived message (see Section 9.3).

Archiving is a method of keeping disk copies of messages that are exchanged between an MTA and its registered Agents and between the MTA and peer MTAs in other routing domains.

By default, Archiving is disabled. If you require disk copies of messages exchanged with a particular registered Agent, or peer MTA in another routing domain, enable Archiving for that particular Agent or peer MTA.

You should be aware that archived message copies reduce the availability of disk space to the MTA and that the process of writing these copies to disk can reduce MTA performance. The MTA provides no automatic management for Archiving. This means that archived messages can accumulate and occupy a large amount of disk space.

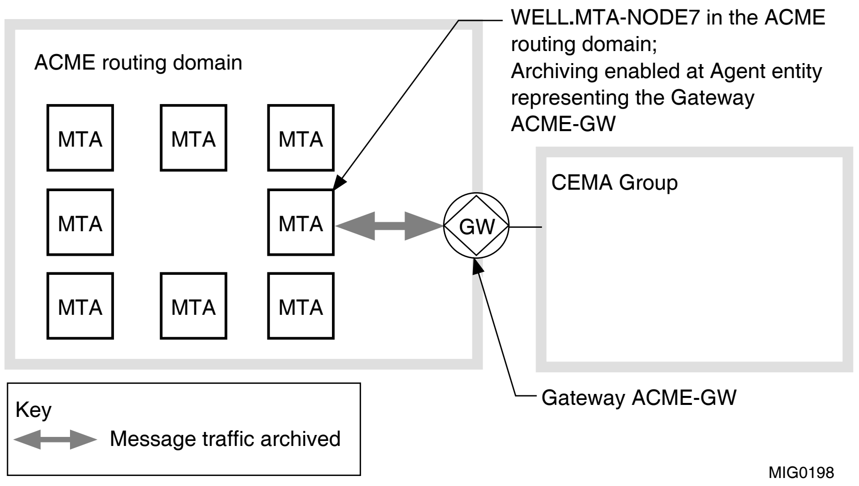
Section 9.1 gives an example of why you might want to archive messages and where in your routing domain you would enable Archiving. Section 9.2 describes how to use Archiving.

9.1 Example of Using Archiving

The messages exchanged between organizations might be considered legally binding documents. Examples are the exchange of electronic data interchange (EDI) documents or the exchange of messages with a Telex Gateway. In this case an organization may want to save copies of such messages by using Archiving at the MTAs that exchange messages with such Gateways or with peer MTAs in other X.400 routing domains.

Figure 9–1 illustrates how ACME Shoe Corporation would use Archiving. In this example ACME exchanges messages which constitute legally binding documents with another organization through a Gateway. Therefore, Archiving is enabled at the relevant Agent entity of the MTA that connects to the Gateway.

Figure 9–1 Archiving Used at a Gateway Connection



WELL.MTA-NODE7 in the ACME routing domain; Archiving enabled at Agent entity representing the Gateway ACME-GW
ACME routing domain
MTA
MTA
MTA
MTA
MTA
MTA
MTA
MTA
GW
CEMA Group
Gateway ACME-GW
Key
Message traffic archived
MIG0198

9.2 How to Tune Archiving

You can tune Archiving as follows:

- Enable or disable Archiving and set the direction of message traffic that is archived (Section 9.2.1).
- Process archived message copies (Section 9.2.2).

9.2.1 Enabling and Disabling Archiving

You enable Archiving for a particular registered Agent or peer MTA in another routing domain using the relevant Agent or Peer MTA entity. When you enable Archiving you also specify the direction of message traffic that is to be archived, as follows:

- Outbound

  Message copies sent from your routing domain through delivery, export or outbound transfer.
- Inbound

  Message copies sent to your routing domain through submission, import or inbound transfer.

- Inbound and outbound

  Message copies sent to and from your routing domain.

Use the SET command as described in the *MTA Module Online Help* to enable or disable Archiving. For example, use the following command to enable or disable Archiving for a particular registered Agent:

```
SET NODE "node-id" MTA Agent "agent-name" ARCHIVE value
```

where `agent-name` is the name of the registered Agent and `value` is one of the following:

- `OUTBOUND`
- `INBOUND`
- `INBOUND AND OUTBOUND`
- `OFF`

9.2.2 Processing Archived Messages

When Archiving is enabled, the MTA creates disk copies of messages that it exchanges with the selected peer MTAs and registered XAPI Agents. These message copies are stored in ASN.1 BER format. The MTA adds a header of 512 bytes to each archived message. See Section 9.3 for a description of the ASN.1 BER in the headers of archived messages.

A Message Decoder tool that decodes messages and displays them as readable text is provided with the MTA. Refer to the appendix describing the operating system specific information for details of how to use the Message Decoder tool.

Tru64 UNIX

Archived message copies are stored in a specific directory. For the location of the Archive directory see Appendix D. The file names of archived messages have the following syntax:

```
2034ECC0A05FCA11860508002307347IACME-GW
```

The filename starts with a unique identifier (UID). The last part of the file name, in the above example the string `ACME-GW`, indicates the name of the registered Agent or peer MTA from which the message was received or to which it was passed.

The one or two letters in the file name before the name of the Agent or Peer MTA (`I` in the above example) indicate how the message entered or left the MTA, where:

`S` indicates submission
`D` indicates delivery

`I` indicates import
`E` indicates export
`TI` indicates inbound transfer
`TO` indicates outbound transfer

♦

OpenVMS

Archived message copies are stored in one or more specific directories. See Appendix E for the logical to use to access the Archive directories. The file names of archived messages have the following syntax:

```
2034ECC0A05FCA11860508002307347TO.EMAC-MTA
```

The file name starts with a unique identifier (UID), which encompasses all characters except the one or two characters before the period. The one or two characters before the period (`TO` in the above example) indicate how the message entered or left the MTA, where:

`S` indicates submission
`D` indicates delivery
`I` indicates import
`E` indicates export
`TI` indicates inbound transfer
`TO` indicates outbound transfer

The file extension following the period, in the above example the string `EMAC-MTA`, indicates the name of the registered Agent or peer MTA from which the message was received or to which it was passed.
♦

Remove archived message copies from the MTA's disk regularly before they occupy too much disk space and affect MTA operation. This is especially important as the MTA does not purge archived messages.

To find out how often to remove archived messages from an MTA's disk, you need to find out how much disk space archived messages occupy. If you are already using Archiving, monitor the size of archived messages in the Archive directory (Tru64 UNIX) or directories (OpenVMS). If you have not used Archiving, you can use the following method to determine the disk space that may be required for your archived messages:

Estimate the number of messages the relevant MTAs would archive and the average size of an archived message. If your MTS is already operational, you can use the following counters to help you determine the number of messages and in the case of a peer MTA connection, the message size:

- Counters of Agent entities:

  MPDUS In
  MPDUS Out

- Counters of Peer MTA entities that are manually configured:

  MPDUS In
  MPDUS Out
  Octets In
  Octets Out

Section 7.2 describes this in more detail.

Multiplying the number of messages with the average message size gives you an indication of the amount of disk space that archived messages can occupy. Check the figures that you have determined against the availability of disk space at each relevant MTA. Set up a method of removing archived messages before they occupy a large amount of disk space and affect MTA operation.

If you receive the System Interface Error event in combination with other events indicating data losses, and if counters indicating such data losses increase suddenly and substantially, you should investigate whether this is due to archived messages occupying too much disk space. If this is the case you must copy archived messages to another storage medium more frequently.

9.3 Archived Message Format

When a MAILbus 400 MTA archives a message, it creates a file which consists of an ASN.1 BER archive header followed by the ASN.1 BER encoding of the message. The archive header is of a fixed size and always occupies the first 512 bytes of an archived message. The ASN.1 BER in the archive header is described in Section 9.3.1. To use the information in Section 9.3.1 you must already be familiar with handling encoded ASN.1 data.

The ASN.1 BER in a message is defined in CCITT Recommendation X.411. Note that a 1992 X.400 message can be encapsulated in an EXTERNAL encoding. The EXTERNAL datatype is defined in CCITT Recommendation X.208 and International Standard ISO 8824.

Use the Message Decoder tool to decode the header of an archived message. See the appendix describing the operating system specific information for the commands to run this tool.

9.3.1 ASN.1 in Archived Message Headers

```
header ::= [PRIVATE 1] IMPLICIT ArchiveMessageHeader;

ArchiveMessageHeader ::= SET {
    1 archive-time INTEGER,
    2 archive-point ArchivePoint,
    CHOICE {
        3 agent-name            [0] IMPLICIT PrintableString,
        4 mta-name              [1] IMPLICIT PrintableString
          }
                              }

ArchivePoint ::= ENUMERATED {
    submission   (0),
    delivery     (1),
    import       (2),
    export       (3),
    transfer-in  (4),
    transfer-out (5)
                            }
```

where

1 `archive-time` indicates when the message was archived. The value is the time in seconds since 00:00:00 Greenwich Mean Time (GMT), January 1, 1970.

2 `archive-point` indicates the process during which the message was archived. The value is either submission, delivery, import, export, transfer-in or transfer-out.

3 `agent-name` indicates the name of the submitting, delivering, importing or exporting Agent. The value is the identifier of the relevant Agent entity.

4 `mta-name` indicates the name of the transferring peer MTA. The value is the identifier of the relevant Peer MTA entity.

10 Message History Logging

This chapter describes how you can tune Message History logging to suit your management requirements.

Use Message History logging to keep information at an MTA about the messages that it has processed. This information can be used to trace messages in your routing domain as described in Section 18.4. By default, Message History logging is disabled; if you require Message History information you must enable Message History logging at the respective MTA. You can access Message History information through the Processed Message entity. You can control how long Message History information remains available at an MTA. Note that Message History information is not kept about probes and reports.

The MTA automatically purges Message History information at regular intervals. However, you should be aware that the process of writing Message History information can reduce MTA performance. Message History information also reduces the disk space available to the MTA.

Section 10.1 gives examples of why you might want to collect Message History information and where in your routing domain you would enable Message History logging. Section 10.2 describes how to use Message History logging.

10.1 Examples of Using Message History Logging

Use the examples in this section as a guide for assessing whether and where to use Message History logging in your routing domain.

10.1.1 Tracing Messages in your Routing Domain

An organization may need Message History information in order to trace a message in its routing domain that did not reach an intended recipient.

Although it is impossible that a message is 'lost' without a corresponding non-delivery report or event, you may want to have history information available to trace the path a message has taken. To have relevant information available, Message History logging could be used at all MTAs in the routing domain. If you decide to enable Message History logging at all MTAs, ensure that each MTA has sufficient disk space available to store Message History information.

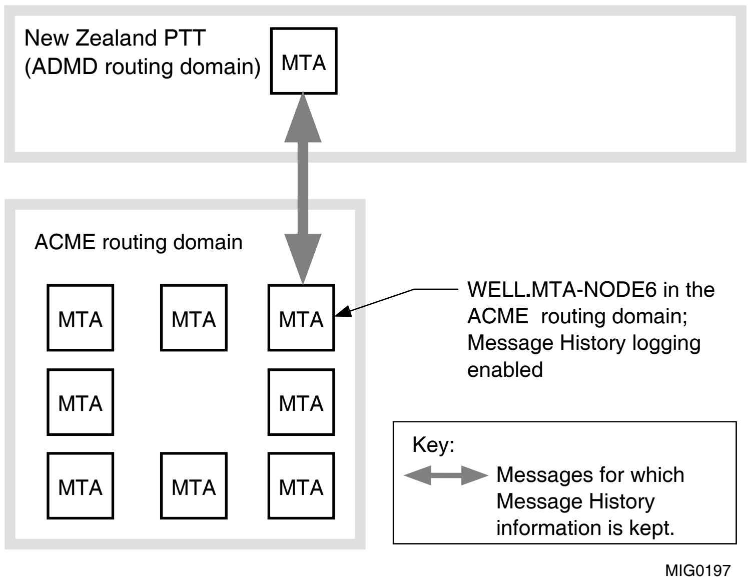
10.1.2 Establishing Whether a Message Was Passed to Another Routing Domain

An organization that connects to other routing domains may need to be able to establish whether a message has been passed to another domain. For example, if a message failed to reach a recipient in another routing domain, you may need to find out whether that message is still in your routing domain.

To have relevant information available, Message History information needs to be collected at the MTAs that exchange messages with Gateways or with peer MTAs in other routing domains.

Figure 10–1 shows that ACME Shoe Corporation uses Message History logging at the boundary MTA connecting to the ADMD.

Figure 10–1 Message History Logging for the ADMD Connection


New Zealand PTT
(ADMD routing domain)
MTA
ACME routing domain
MTA
MTA
MTA
MTA
MTA
MTA
MTA
MTA
WELL.MTA-NODE6 in the ACME routing domain; Message History logging enabled
Key:
Messages for which Message History information is kept.
MIG0197

10.2 How to Tune Message History Logging

You can tune Message History logging as follows:

- Enable or disable Message History logging at an MTA (Section 10.2.1).
- Change the interval at which Message History information is purged (Section 10.2.2).

10.2.1 Enabling, Disabling and Viewing Message History Logging

If you require Message History information, for example to trace messages as described in Section 18.4, you must enable Message History logging.

When Message History logging is enabled at an MTA, this MTA retains the following information about each message (excluding probes and reports) that it processes. This information is held in the Processed Message entity:

- The message identifier
- The O/R addresses of all recipients
- Information indicating whether the message was expanded from a distribution list (the information includes the O/R addresses of the recipients in the distribution list) or redirected (the information contains the O/R address to which the message was redirected)
- The state of the message for each recipient that the MTA knows about
- The next destination of the message

You can display the Message History logged by an MTA using the Processed Message entity. See Section 18.4 for an example output of the Processed Message entity.

Use the following command to enable or disable Message History logging at an MTA:

```
SET NODE "node-id" MTA MESSAGE HISTORY STATE value
```

where *value* is either `ON` (enable) or `OFF` (disable).

10.2.2 Changing the Message History Purge Interval

Message History information is stored in a specific workspace. For the location of this workspace, refer to the appendix describing the operating system specific information. The Message History Purge Interval attribute automatically purges Message History information after a certain period of time. Unless modified, the setting of this attribute is seven days.

For purging to take place, the MTA must be enabled and the purge interval must not be zero (0-00:00:00). When the Message History Purge Interval is zero, purging is disabled.

Before you change the Message History Purge Interval at an MTA, decide what service you want to offer your MHS users; for example, how long after a message was submitted do you want to be able to investigate its ‘loss’. Also, find out how much disk space your current Message History files use. You must then find a compromise between the degree of service you want to offer and the amount of disk space that you have available for Message History files.

To ensure that Message History information is reliable and consistent, set the Message History Purge Interval to the same value at all MTAs where Message History logging is enabled.

Use the following command to modify the Message History Purge Interval:

```
SET NODE "node-id" MTA MESSAGE HISTORY PURGE INTERVAL d-hh:mm:ss
```

where *d-hh:mm:ss* is the desired purge interval in binary relative time. Only the value you enter for `d` is used; that is, the value you supply for the number of days.

If you receive the System Interface Error event in combination with other events indicating data losses, and if counters indicating such data losses increase suddenly and substantially, investigate whether this is due to Message History information occupying too much disk space. If this is the case you must either purge Message History information more frequently or otherwise ensure that sufficient disk space remains available for the MTA to operate.

11 Content Information and IPM Bodypart Converters

The MAILbus 400 MTA can transfer messages of any content length and content type, containing any data format. In addition, MAILbus 400 MTAs can convert bodyparts in messages of the interpersonal messaging (IPM) content type, if suitable converters are available. Unless you specify otherwise, MAILbus 400 MTAs deliver all messages regardless of their content length, and the data formats or IPM bodyparts they carry. If you want your Agents to receive only messages of a certain content type and content length, containing only certain IPM bodyparts or data formats, you need to indicate such restrictions specifically. Messages are then delivered or exported on the basis of these restrictions.

Usually, it is the capability of your Agents that determines the restrictions that apply. However, users may also have preferences with regard to the messages that they receive. You specify Agent restrictions and user preferences by supplying information in the directory, where it can be shared by all MTAs in your routing domain. This information is called **content information** and is held in the directory as attributes of the O/R address or O/R addresses that a particular Agent serves. MAILbus 400 MTAs use this content information to decide whether to deliver or export a message and whether to convert bodyparts in an IPM. You enter content information in the directory as object identifiers. An **object identifier** is a sequence of numbers that uniquely identifies an object, in this case a content type, IPM bodypart or data format.

Section 11.1 describes content information in more detail and lists the object identifiers for the content types.

Section 11.2 describes the content types that are most likely to occur in an MTS.

Section 11.3 describes encoded information types (EITs) and the data formats and IPM bodyparts that are most likely to occur in an MTS.

Section 11.4 describes the bodypart converters that are supplied with the MAILbus 400 MTA.

Section 11.5 provides some examples and guidelines for entering content information in the directory.

Chapter 12 describes how to assign object identifiers to proprietary content types, data formats or bodyparts. Chapter 12 also describes how to define or create Bodypart entities for proprietary bodyparts and how to integrate a proprietary converter with the MTA.

11.1 Content Information

Content information for an O/R address consists of the following:

- One or more identifiers for the message content types that an Agent supports.
- A value for the maximum content length that an Agent supports.
- Identifiers for the data formats or IPM bodyparts within the supported content types. These identifiers are called **encoded information type (EIT)** identifiers.

When routing a message to a recipient, the MTA compares information held in the message envelope with the content information in the recipient's O/R address entry in the directory, and decides what to do with the message. To prevent an MTA from passing messages to an Agent that the Agent cannot support, you must specify which content length, content types, data formats or IPM bodyparts the Agent can handle. Otherwise, the following situations can occur:

- Messages of a content type that an Agent does not support can be delivered. A recipient's User Agent might not be able to display a message if it is of an unsupported content type.
- Data formats or IPM bodyparts that an Agent does not support can be present in a message. A recipient's User Agent might not be able to display a certain data format or IPM bodypart.
- Messages of content lengths that an Agent does not support can be delivered. A recipient's Agent might not be able to handle a message that is too large.

11.1.1 Effects of Specifying Content Information

Specifying content information according to Agent restrictions and user preferences (if applicable) has the following effects:

- The MTA only delivers or exports messages of content types that are specified in the content information.

  Messages of other content types are not delivered or exported, and the MTA sends a non-delivery report to the message originator, if possible.

- The MTA does not deliver or export messages whose content length exceeds the value supplied in the content information.

  The MTA rounds up the content length of a message to the nearest Kilobyte and compares it with the value supplied as content information. If the message content is too big, the MTA does not deliver or export the message. The MTA then sends a non-delivery report to the message originator, if possible.

  Note that this applies to the content size of unconverted messages only, as the MTA does not recalculate the content size of a message after a conversion.

- The MTA attempts to convert unacceptable bodyparts in IPMs.

  This applies when the IPMS content type is specified as acceptable in the content information.

  When an MTA receives an IPM with bodyparts whose EITs are not listed in the recipient's content information, the MTA attempts to convert these bodyparts to a bodypart whose EIT is listed in the content information. Section 11.3 describes EITs in more detail. If an MTA cannot perform the required conversions, it transfers the unconverted IPM to the next MTA on the IPM's route. See Section 3.8 for more information about the conversion process. The converters provided with the MTA are described in Section 11.4.

- The MTA does not deliver or export messages containing unacceptable EITs.

  This applies regardless of the content types that are specified as acceptable in the content information.

  If data formats contained in a message do not match the data formats whose EITs are listed in the content information, the MTA does not deliver or export the message. The MTA sends a non-delivery report to the message originator, if possible.

Each MTA that handles a message compares the message format with the applicable content information and, if the message is an IPM and unacceptable bodyparts are present, attempts to convert these bodyparts. It is the last MTA on a message's route (that is, the exporting or delivering MTA) that decides whether a message is in the correct format and can be delivered or exported. An MTA that is not last on a message's route (that is, a transferring MTA) transfers a message to another MTA, even if the message format is unacceptable. This is true for messages that are transferred between MTAs within your routing domain and between a boundary MTA and a peer MTA in another X.400 routing domain. Transferring an unacceptable message instead of non-delivering it maximizes the probability that an unacceptable bodypart in an IPM can be converted by another MTA on the route.

You need to decide whether to specify content information in your O/R addresses to reflect your Agents' capabilities. This depends on how likely it is that your Agents will receive messages that are unacceptable to them and how important it is to you or your users that this does not occur. If you do not specify any content information, the MTA delivers and exports all messages, irrespective of their length, content type, and the data formats or bodyparts contained in them.

You also need to decide whether to represent restrictions that you are aware of in other X.400 routing domains to which you are connecting, even though boundary MTAs do not make delivery or non-delivery decisions. Specifying content information for users in other X.400 routing domains is important, particularly where you have an X.400 routing domain based on the 1984 MHS Standards connecting to your MAILbus 400 MTA routing domain (which is based on the 1992 MHS Standards).

Chapter 6 describes in detail the actions of the MTA when downgrading a message for a 1984 routing domain.

You can add content information at any point in your O/R address hierarchy, As an example, a group of users in another routing domain is connected to your routing domain through a Gateway, which supports only messages of one content type. In this case, you could place the applicable content information at the partial O/R address entry in the directory that represents these users.

See Section 11.2 for information on how to specify acceptable content types and content length, and Section 11.3 for information on how to specify acceptable EITs. Section 11.4 describes the bodypart converters supplied with the MTA and Section 11.5 describes how to enter this information in the directory using entities of the MTS module.

11.2 Content Type and Content Length

To identify the content types and the maximum content length that an Agent supports, refer to the documentation supplied with the Agent.

Some Agents can accept messages of any content type, but actually process only some content types effectively. Others can accept and process only one particular content type. As an example, mail User Agents conforming to the 1984 MHS Standards typically accept and support the 1984 IPMS content type only, while mail User Agents conforming to the 1988 MHS Standards or 1992 MHS Standards must accept and support both the 1984 and the 1992 IPMS content types. This is because the 1984 IPMS content type is a subset of the 1992 IPMS content type.

Note

The definition of the IPMS (or interpersonal messaging) content type has not changed significantly between the 1988 and the 1992 MHS Standards. Therefore, the terms “1992 IPMS content type” or “interpersonal messaging 1992” when used in the MTA documentation include the 1988 definition of the IPMS content type.

Section 11.5 provides some guidelines for the content information that is applicable to different mail systems.

The content types that are most likely to occur in an MTS are listed in Table 11–1. This table also lists the object identifiers assigned by HP to the individual content types. Enter these object identifiers in the O/R address entries of your users as required.

Table 11–1 Content Types

Content type	Object Identifier	Description
Any content type	{1 3 12 2 1011 5 5 0 0}	Allows any content type to be delivered.
Unidentified	{1 3 12 2 1011 5 5 0 1 0}	Used by bilateral agreement between users of an MTS.
Electronic Data Interchange	{1 3 12 2 1011 5 5 0 1 35}	Used for documents of the EDI content type.
Interpersonal messaging 1984	{1 3 12 2 1011 5 5 0 1 2}	IPMS content type as defined by the 1984 MHS Standards. This setting causes the MTA to downgrade 1992 interpersonal messages to 1984 IPMS format. Chapter 6 describes in detail how the MTA downgrade a message to 1984 IPMS format.
Interpersonal messaging 1992 Select one of the following:		
1992 Externally Defined IPMS	{1 3 12 2 1011 5 5 0 1 22}	Use this content type for users who use applications such as ALL-IN-1™, MailWorks and the MAILbus 400 Message Router Gateway. This setting represents the IPMS content type as defined by the 1992 MHS Standards and includes the interpersonal messaging 1984 content type setting. In this setting, File Transfer bodyparts (FTBPs) in 1992 interpersonal messages are translated to Externally Defined bodyparts. An example of an application generating File Transfer bodyparts is Microsoft® Exchange Server. See Section 11.2.1 for information on bodypart translation and how you can modify it according to your requirements.

(continued on next page)

Table 11–1 (Cont.) Content Types

Content type	Object Identifier	Description
1992 File Transfer IPMS	{1 3 12 2 1011 5 5 0 1 22 1}	Use this content type for users who use an application such as Microsoft Exchange Server. This setting represents the IPMS content type as defined by the 1992 MHS Standards and includes the interpersonal messaging 1984 content type setting. In this setting, Externally Defined bodyparts in 1992 interpersonal messages are translated to File Transfer bodyparts. Examples of applications that generate Externally Defined bodyparts are ALL-IN-1 and MailWorks. See Section 11.2.1 for information on bodypart translation, and how you can modify it according to your requirements.
1992 IPMS Passthrough	{1 3 12 2 1011 5 5 0 1 22 0}	Use this setting if you want Externally Defined bodyparts and File Transfer bodyparts in 1992 interpersonal messages to be passed through untranslated. This setting represents the IPMS content type as defined by the 1992 MHS Standards and includes the interpersonal messaging 1984 content type setting.

It is possible that you are using proprietary content types in your routing domain for which the MTA documentation cannot supply unique object identifiers. If these content types are described by object identifiers, you can use these object identifiers in the content information of your O/R addresses, otherwise, see Chapter 12.

11.2.1 Bodypart Translation

Some data formats can be transferred in 1992 interpersonal messages either as Externally Defined bodyparts or as File Transfer bodyparts. While both types of bodypart can carry the same data, they use different means of identifying the data. Most recipients' User Agents can only understand one type of bodypart, either File Transfer or Externally Defined, and therefore the bodyparts need to be translated. An example of an application that generates File Transfer bodyparts is Microsoft Exchange Server. Examples of applications that generate Externally Defined bodyparts are ALL-IN-1, MailWorks and the MAILbus 400 Message Router Gateway.

You specify whether and how bodypart translation takes place by specifying one of the interpersonal messaging 1992 content types listed in Table 11–1 in a user's content information in the directory.

When translating a bodypart, the MTA changes the information that *identifies* the bodypart, it does not modify the data contained in the bodypart. This enables a recipient's User Agent to recognize the type of bodypart in a message and deal with it appropriately. (This is different from bodypart conversion, where the MTA changes the format of the data contained in a bodypart.)

To translate between Externally Defined bodyparts and File Transfer bodyparts in 1992 interpersonal messages, the MTA uses a bodypart mapping table. This table contains the translations defined for Externally Defined bodyparts and File Transfer bodyparts.

You can modify bodypart translation if the way the MTA translates between bodyparts does not suit the requirements in your routing domain. To change how the MTA translates bodyparts you need to change the information in the bodypart mapping table. Instructions on how to change the bodypart mapping table are provided in the bodypart mapping table itself.

The bodypart mapping table is located at
/var/mta/mta_bp_map_table.txt (Tru64 UNIX) or
SYS$COMMON:[MTA]MTA$BP_MAP_TABLE.TXT (OpenVMS).

11.3 Encoded Information Types (EITs)

For each content type that you specify you need to list EITs to indicate which data formats within that content type, or which bodyparts within IPMs, can be received by your Agents. To identify which data formats or IPM bodypart types an Agent supports, see the documentation supplied with the Agent.

Conversion of IPM bodyparts by the MTA is dependent on which EIT identifiers are supplied in the content information of an O/R address. You enter the applicable EITs as object identifiers. The order in which EITs are listed represents the order in which you want an MTA to prioritize the choice of conversion. HP recommends that you list formats with more features before formats with fewer features. In doing so, you reduce the risk of losing information in the conversion process.

If you want an MTA to attempt to convert unsupported IPM bodyparts, but to still deliver the message if no conversion is possible, add the value for “any EIT” to the list of EITs. You can place this value anywhere in the list of applicable EITs without affecting the order of preference in which conversions are performed.

Note that you can set up content information for an O/R address to allow the Agent to receive the combination of 1984 IPMS content type and EITs defined for Externally Defined bodyparts which are strictly associated with the 1992 IPMS content type. In this case, the MTA will downgrade these bodyparts as described in Chapter 6.

You can also set up content information for an O/R address such that messages can be transferred across an intermediate routing domain based on the 1984 MHS Standards if that routing domain cannot transfer the 1992 IPMS content type; see Section 11.5.7.

The data formats and IPM bodypart types that are most likely to occur in an MTS are listed in Table 11–2. This table also lists the EITs that you need to enter to represent these data formats or IPM bodyparts in your O/R address entries in the directory.

The MTA’s startup script creates Bodypart entities for each of the bodyparts listed in Table 11–2. The MTA startup script executes an additional script which creates additional Externally Defined Bodypart entities that might be required for the User Agent or Gateway products you are using. For the location of the MTA’s startup script see the appendix describing the operating system specific information.

Table 11–2 Data Formats and IPM Bodyparts

Data Format or Bodypart	EITs	Description
Any EIT	{1 3 12 2 1011 5 5 1 0}	Allows delivery of any data format or bodypart.
ISO 6937[1]	{1 3 12 2 1011 5 5 1 1 11}	Contains text created using the ISO 6937 character set.

[1]Bodypart defined in International Standard ISO DIS 9065 (the obsolete ISO standard corresponding to CCITT 1984 Recommendation X.420). See Appendix A of this guide.

(continued on next page)

Table 11–2 (Cont.) Data Formats and IPM Bodyparts

Data Format or Bodypart	EITs	Description
ODIF[2] (any Document Application Profile (DAP))	Use the EITs for Externally Defined ODIF Q111, ODIF Q112 and ODIF Q121 specified further on in this table.	Bodypart in 1984 encoding that contains a document in Open Document Architecture (ODA). ODA documents can also occur in a 1992 Externally Defined encoding. See further on in this table.
USA Nationally Defined	{1 3 12 2 1011 5 5 2 0 310}	Contains a bodypart defined by the Stable Implementation Agreements[2], and whose semantics and syntax are defined by registration within the U.S.A.
IA5Text[3]	{2 6 3 4 2}	Contains text created using the IA5 international reference version (IRV) character set. IA5 text characters are similar to ASCII text characters.
Voice[3]	{2 6 3 4 7}	Contains digitized speech.
G3Fax[3]	{2 6 3 4 3}	Contains data in an encoding necessary for document transmission using Group 3 facsimile apparatus.
G4Class1[3]	{2 6 3 4 4}	Contains data in an encoding necessary for document transmission using Group 4 Class 1 facsimile apparatus.
Teletex[3]	{2 6 3 4 5}	Contains data suitable for transmission using teletex terminal equipment.
Videotex[3]	{2 6 3 4 6}	Contains videotex data.
Encrypted[3]	{1 3 12 2 1011 5 5 2 0 8}	Contains another bodypart in encrypted form.
Message[3]	No EIT required, the MTA delivers on the basis of bodyparts contained in the forwarded message.	Bodypart that contains an IPM, enclosed in another, to represent forwarded messages.

[2]Bodypart defined in the Stable Implementation Agreements for Open Systems Interconnection Protocols, Version 7, Edition 1, December 1993, Chapter 7. See Appendix A of this guide.

[3]Bodypart defined in CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7. See Appendix A of this guide.

(continued on next page)

Table 11–2 (Cont.) Data Formats and IPM Bodyparts

Data Format or Bodypart	EITs	Description
Mixed-mode[3]	{2 6 3 4 9}	Contains a mixed mode document that contains characters and raster graphics.
Bilaterally Defined[3]	{1 3 12 2 1011 5 5 2 0 14}	Contains data whose semantics and syntax are agreed between the originator of the IPM and all of its potential recipients.
Nationally Defined[3]	{1 3 12 2 1011 5 5 2 0 7}	Contains data whose semantics and syntax are agreed on a national basis.
Externally Defined[3], can contain for example:		Used for bodyparts that do not fit into any of the other categories.
DDIF[4]	{1 3 12 1011 1 3 1}	Contains a document in Digital Document Interchange Format (DDIF), as used in HP's compound document architecture, CDA.
DECdx[4]	{1 3 12 1011 1 3 8}	Contains data in DECdx™ format. DECdx is an intermediate data format generated by HP Gateways (such as the Message Router/P and Message Router/S Gateways) and User Agents.
DEC MCS[4]	{1 3 12 2 1011 5 1 209}	Contains characters from the DEC Multi-national character set (MCS).
DTIF[4]	{1 3 12 1011 1 3 3}	Contains a document in Digital Tabular Interchange Format (DTIF), as used in HP's compound document architecture, CDA.
DOTS[4]	{1 3 12 1011 1 3 2}	Contains a document in the Data Object Transport Syntax (DOTS) mail interchange format. DOTS is used to encapsulate data elements that have links between them, for example, a DDIF document and its references to external documents.

[3]Bodypart defined in CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7. See Appendix A of this guide.

[4]Bodypart defined by HP.

(continued on next page)

Table 11–2 (Cont.) Data Formats and IPM Bodyparts

Data Format or Bodypart	EITs	Description
Postscript[4]	{1 3 12 1011 1 3 6}	Contains a document in the PostScript® interchange and presentation format.

ODIF Q111[5]	{2 8 1 0 1}
	{0 0 20 502 0}
		Contains a document conforming to ODA document application profile (DAP) Q111.

ODIF Q112[5]	{2 8 1 0 1}
	{1 3 16 2 6 0 1}
		Contains a document conforming to ODA document application profile (DAP) Q112.

ODIF Q121[5]	{2 8 1 0 1}
	{1 3 16 2 6 0 2}
		Contains a document conforming to ODA document application profile (DAP) Q121.

Message Router Text[4,6]	{1 3 12 2 1011 5 1 210}	Assumed to contain characters from the DEC Multi-national character set (MCS).
WPS-PLUS[4]	{1 3 12 1011 1 3 7}	Contains a document in WPS-PLUS™ format.
SDK	{1 3 12 2 1011 5 1 184}	Super DEC Kanji Text bodypart.
SJIS	{1 3 12 2 1011 5 1 185}	Shift JIS Text bodypart.
jpbody88	{1 2 392 6 1 4 0}	Bodypart defined by INTAP[7] as Extended JP1. This bodypart is for IPMS content type Interpersonal Messaging 1992.

[4]Bodypart defined by HP.

[5]Bodypart defined in International Standard ISO 8613 and related profiles Q111, Q112 or Q121. See Appendix A of this guide.

[6]This bodypart can be used for a bodypart other than DEC MCS, for example, a National Replacement Character set; see the MAILbus 400 Message Router Gateway documentation for details.

[7]INTAP is Interoperability Technology Association for Information Processing Japan

(continued on next page)

Table 11–2 (Cont.) Data Formats and IPM Bodyparts

Data Format or Bodypart	EITs	Description
General text[3]		Can carry any character set data registered in “International Register of ISO Coded Character Sets” published by ISO. Use the object identifiers stem {1 0 10021 7 1 0 n}, where n is the integer defined for the appropriate character set(s) in above-mentioned ISO publication.
Examples of General Text:		

ISO Latin 1	{1 0 10021 7 1 0 6}
	{1 0 10021 7 1 0 100}
	{1 0 10021 7 1 0 1}
	{1 0 10021 7 1 0 77}
		Contains characters from the ISO Latin 1 character set.

T.61 Latin	{1 0 10021 7 1 0 102}
	{1 0 10021 7 1 0 103}
	{1 0 10021 7 1 0 106}
	{1 0 10021 7 1 0 107}
		Contains characters from the T.61 Latin character set.

IA5	{1 0 10021 7 1 0 2}
	{1 0 10021 7 1 0 1}
		Contains characters from the IA5 character set.

jpbody84	{1 3 12 2 1011 5 5 2 0 440}	Bodypart defined by INTAP[7] as JPBodyParts. This bodypart is for IPMS content type Interpersonal Messaging 1984.

[3]Bodypart defined in CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7. See Appendix A of this guide.

[7]INTAP is Interoperability Technology Association for Information Processing Japan

It is possible that you are using proprietary bodyparts or data formats in your routing domain for which the MTA documentation cannot supply unique object identifiers. In this case see Chapter 12.

11.4 Bodypart Converters Supplied with the MTA

A MAILbus 400 MTA is supplied with several converter images that enable it to convert messages from one bodypart type to another. For example, it can automatically convert a Teletex bodypart to a General Text bodypart containing ISO Latin 1 characters if requested.

The MTA's startup script contains commands that create a Converter entity for each of the converters supplied with the MTA. A Converter entity does the following:

- Identifies a particular converter image.
- Identifies the Bodypart entities that describe the input bodypart to the converter and the output bodypart from the converter.
- Specifies whether data could be lost during conversion.
- If required, specifies the sequence of conversions needed to convert the input bodypart to the output bodypart.

Table 11–3 lists a selection of Converter entities created by the MTA's startup script. A complete list of the Converter entities can be found in the MTA startup script. Refer to the appendix describing the operating specific information for the location of the MTA's startup script. Table 11–3 also lists the Bodypart entities that these Converter entities reference.

The Lossy column in Table 11–3 attribute indicates whether or not data could be lost during conversion. This attribute has the value TRUE or FALSE. TRUE means that data could be lost during the conversion.

Table 11–3 Examples of Converter Entities Supplied with the MTA

Converter entity	Input (Source Bodypart entity)	Output (Target Bodypart entity)	Lossy
"ia5tolatin1"	"ia5text" (IA5 text)	"isolatin1" (ISO Latin 1)	False
"latin1tot61"[1]	"isolatin1" (ISO Latin 1)	"teletex" (Teletex)	False
"latin1togeneralia5"	"isolatin1" (ISO Latin 1)	"generaltextia5" (General Text containing IA5 characters)	True
"generaltoia5"	"generaltextia5" (General Text containing IA5 characters)	"ia5text" (IA5 text)	False
"generalt61tolatin1"	"generaltextt61" (General Text containing T.61 Latin characters)	"isolatin1" (ISO Latin 1)	True
"t61togeneral"	"teletex" (Teletex)	"generaltextt61" (General Text containing T.61 Latin characters)	False
"generaltot61"	"generaltextt61" (General Text containing T.61 Latin characters)	"teletex" (Teletex)	False

[1]T.61 is the name used in the 1984 MHS Standards to describe a Teletex bodypart.

(continued on next page)

Table 11–3 (Cont.) Examples of Converter Entities Supplied with the MTA

Converter entity	Input (Source Bodypart entity)	Output (Target Bodypart entity)	Lossy
"t61tolatin1"	"teletex" (Teletex)	"isolatin1" (ISO Latin 1)	True
"latin1toia5"	"isolatin1" (ISO Latin 1)	"ia5text" (IA5 text)	True
"iso6937tolatin1"	"iso6937" (ISO 6937)	"isolatin1" (ISO Latin 1)	True
"latin1toiso6937"	"isolatin1" (ISO Latin 1)	"iso6937" (ISO 6937)	False
"externaldeftobilatdef"	"externallydefined" (Externally Defined)	"bilaterallydefined" (Bilaterally Defined)	False[4]
"externaldeftoposte"[3]	"externallydefined" (Externally Defined)	"bilaterallydefined" (Bilaterally Defined)	False[4]
"decdxtolatin1"	"decdx" (DECdx)	"latin1" (ISO Latin 1)	True
"wpsplustolatin1"	"wpsplus" (WPS-PLUS)	"latin1" (ISO Latin 1)	True
"decmcstolatin1"	"decmcs" (DEC Multi-national character set)	"latin1" (ISO Latin 1)	True
"latin1todecmcs"	"isolatin1" (ISO Latin 1)	"decmcs" (DEC Multi-national character set)	True
"mrtexttolatin1"	"mrtext" (Message Router Text)	"isolatin1" (ISO Latin 1)	True
"latin1tomrtext"	"isolatin1" (ISO Latin 1)	"mrtext" (Message Router Text)	True
"j84tosdk"[5]	"jpbody84" (JPBodyPart)	"sdk" (Super DEC Kanji)	False
"sdktoj84"[5]	"sdk"(Super DEC Kanji)	"jpbody84" (JPBodyPart)	True
"j88tosdk"[5]	"jpbody88"(Extended JP1)	"sdk" (Super DEC Kanji)	False
"sdktoj88"[5]	"sdk" (Super DEC Kanji)	"jpbody88"(Extended JP1)	True
"sjistosdk"[5]	"sjis" (Shift JIS)	"sdk"(Super DEC Kanji)	False
"sdktosjis"[5]	"sdk" (Super DEC Kanji)	"sjis"(Shift JIS)	True

[3]This entity is not registered as part of the MTA startup, see Section 11.4.1.1

[4]The descriptive data in the bodypart is lost, but no message data in the bodypart is lost.

[5]In order to perform this conversion, you will need to install an additional software subset IOSJPBASEnnn, where n is a number. This subset is part of Tru64 UNIX Japanese Support. You also need to remove the comment character (!) from the bodypart converter definition in the MTA startup script. The IOSJPBASEnnn subset is only available on the Tru64 UNIX platform.

If you do not want an MTA to use one or more of its converters, edit the MTA's startup script and insert a comment (`!`) at the start of the relevant create command. For the location of the MTA's startup script, refer to the appendix describing the operating system specific information.

Note

It is recommended that you do not remove a create command for a Bodypart entity from the MTA's startup script.

11.4.1 Converting Externally Defined and File Transfer Bodyparts to Bilaterally Defined Bodyparts

The MTA provides a converter that converts Externally Defined bodyparts and File Transfer bodyparts to Bilaterally Defined bodyparts. This converter is called "externaldeftobilatdef".

Bilaterally Defined bodyparts are also known as Unidentified, binary bodyparts or Bodypart 14. Bilaterally Defined bodyparts are defined in the 1984 MHS Standards and contain unidentified data. Bilaterally Defined bodyparts are typically used for exchanging messages where the same application is used to send and receive the message. An example of the type of bodypart that might be exchanged between PCs that use a Gateway, or MTA, based on the 1984 MHS Standards, is the Microsoft® Excel® V3.0 Macro bodypart.

Externally Defined bodyparts and File Transfer bodyparts are defined in the 1992 MHS Standards and can contain the same types of data as Bilaterally Defined bodyparts. However, Externally Defined and File Transfer bodyparts also include a description of the data in the bodypart, for example, an Externally Defined bodypart containing Microsoft Excel Version 3.0 data can be identified as a "Microsoft Excel Version 3.0" bodypart.

When an MTA converts Externally Defined bodyparts or File Transfer bodyparts to Bilaterally Defined bodyparts, it removes the descriptive information. This means that the receiving Agent must be able to identify the type of information contained in the bodypart once the descriptive data has been removed.

The Externally Defined/File Transfer bodypart to Bilaterally Defined bodypart converter complements the conversion actions performed by the MTA when downgrading messages for routing domains based on the 1984 MHS Standards. It is provided so users with 1984 MHS Standard based User Agents that support Bilaterally Defined bodyparts can receive Externally Defined or File Transfer bodyparts, converted to Bilaterally Defined bodyparts. To achieve this, put the EIT for the Bilaterally Defined bodypart at the end of the list of acceptable EITS in the users' content information.

Section 11.5 provides examples of entering content information for this and other converters in the directory using the entities of the MTS module.

11.4.1.1 Interworking Between MailWorks Server for Tru64 UNIX and the Poste/X.400 Gateway

If your routing domain has both DEC MailWorks for UNIX® and the Poste/X.400 Gateway User Agents, you can use an alternative Externally Defined to Bilaterally Defined bodypart converter supplied with the MAILbus 400 MTA. This converter is called “externaldeftoposte” and is commented out in the MTA’s startup script.

If you remove the comments (!) from the applicable commands in the MTA’s startup script, this converter converts an Externally Defined bodypart to a Poste/X.400 Gateway specific Bilaterally Defined bodypart. A Poste/X.400 Bilaterally Defined bodypart contains an encoded header that requires the MTA to provide a different conversion.

This converter can provide MailWorks Server for Tru64 UNIX and the Poste/X.400 Gateway with a better service. The MailWorks Server for Tru64 UNIX documentation describes when to use this converter.

Note

You cannot convert between Externally Defined bodyparts and both Bilaterally Defined and Poste/X.400 specific Bilaterally Defined bodyparts within the same routing domain. If you register both converters within the same routing domain, recipients can receive the wrong type of Bilaterally Defined bodypart.

11.5 Entering Content Information in the Directory

The following sections provide examples of the content information that you might choose to assign to O/R address entries in the directory.

You can also specify content information that is specific to a particular form of O/R address, using the Mnemonic Content Information, Postal Content Information, Numeric Content Information, and Terminal Content Information attributes. See the *MTS Module Online Help* for more information about how to enter specific content information for an O/R address entry.

11.5.1 Entering Content Information in the Directory for Individuals Using a 1984 Based User Agent

The following is an example of the Content Information that you might provide for individuals who are served by User Agents based on the 1984 MHS Standards:

- Content Type

  This is Interpersonal messaging 1984.

- Encoded Information Types (EITs)

  This is the list of EITs applicable to the user, or group of users, served by the User Agent.

  It is important that you list the EITs in order of preference or priority. You are advised to order the EITs such that those EITs with more features appear in the EIT list before those with fewer features. This reduces the possibility of losing information during conversion.

  If you intend to specify Bilaterally Defined in your list of EITs, make sure that you specify Bilaterally Defined last in the list of EITs.

- Content Length

  Whatever is applicable.

The following is an example O/R address entry for an individual served by a User Agent based on the 1984 MHS Standards. Note that text following an exclamation mark (`!`) is not part of the command; it is added to explain the object identifiers in the command:

```
SET MTS "/MTS=ACME" ORADDRESS -
    "C=NZ;A=NZ-PTT;P=ACME;O=ACME;OU1=WELL;CN=Sue Lyn" -
    CONTENT INFORMATION [MAXIMUM CONTENT LENGTH = 1000, -
    CONTENT TYPES = -
    ("{1 3 12 2 1011 5 5 0 1 2}"), -  ! Interpersonal messaging 1984
    ENCODED INFORMATION TYPES = ( -   !
    "{2 6 3 4 5}", -                  ! Teletex
    "{1 3 12 2 1011 5 5 1 1 11}", -   ! ISO 6937
    "{2 6 3 4 2}", -                  ! IA5Text
    "{1 3 12 2 1011 5 5 2 0 14}")]    ! Bilaterally Defined
```

11.5.2 Entering Content Information in the Directory for Individuals Using a 1992 Based User Agent

The following is an example of the Content Information that you might provide for individuals who are served by User Agents based on the 1992 MHS Standards:

- Content Type

  This is 1992 Externally Defined IPMS, where any File Transfer bodypart is translated into an Externally Defined bodypart.

- Encoded Information Types (EITs)

  Recipients with 1992 based User Agents should be able to receive any type of bodypart. If this is the case, you need only include the “Any” EIT as the preferred EIT. Refer to Section 11.3 for details of using the “Any” EIT.

  If individual recipients have preferred EITs, it is important that you list the EITs in order of preference or priority. You are advised to order the EITs such that those EITs with more features appear in the EIT list before those with fewer features. This reduces the possibility of losing information during conversion. In particular, you are recommended not to use the Bilaterally Defined EIT. If you do use the Bilaterally Defined EIT, make sure that you add any Externally Defined EITs to the list of preferred EITs before the Bilaterally Defined EIT. If you do not specify the Externally Defined bodyparts explicitly, all Externally Defined bodyparts are converted to Bilaterally Defined bodyparts, and this might not be what you intended.

- Content Length

  Whatever is applicable.

The following is an example O/R address entry for an individual served by a User Agent based on the 1992 MHS Standards. Note that text following an exclamation mark (`!`) is not part of the command; it is added to explain the object identifiers in the command:

```
SET MTS "/MTS=ACME" ORADDRESS -
    "C=NZ;A=NZ-PTT;P=ACME;O=ACME;OU1=WELL;CN=Sue Lyn" -
    CONTENT INFORMATION [MAXIMUM CONTENT LENGTH = 1000, -
    CONTENT TYPES = ( -
    "{1 3 12 2 1011 5 5 0 1 22}"), -  ! 1992 Externally Defined IPMS
    ENCODED INFORMATION TYPES = ( -   !
    "{1 3 12 2 1011 5 5 1 0}")]       ! "Any" EIT
```

11.5.3 Entering Content Information for the MAILbus 400 SMTP Gateway

The following is an example of the content information that you might choose to provide for individuals in an Internet network who are served by the HP MAILbus 400 SMTP Gateway Version 2.0 or later:

- Content Type

  This is 1992 Externally Defined IPMS, where any File Transfer bodypart is translated into an Externally Defined bodypart.

- Encoded Information Types (EITs)

  The EITs applicable to an Internet network served by the HP MAILbus 400 SMTP Gateway:

  - General text ISO Latin 1
  - ISO 8859-1 to -9
  - General Text IA5
  - IA5Text
  - DDIF
  - any

- Content Length

  Whatever is applicable for the User Agents. The SMTP Gateway itself has no content length restrictions.

The following is an example of a directory entry for an individual served by the HP MAILbus 400 SMTP Gateway.

```
SET MTS "/MTS=ACME" ORADDRESS -
 "C=NZ; A=NZ-PTT; P=SANDS; O=SANDS; OU1=NORTH; CN=Sue Lyn" -
 CONTENT INFORMATION [MAXIMUM CONTENT LENGTH = 0, -
 CONTENT TYPES = ( -
 "{1 3 12 2 1011 5 5 0 1 22}"),-  ! 1992 Externally Defined IPMS
 ENCODED INFORMATION TYPES = ( -  !
 "{1 0 10021 7 1 0 6}", -         !
 "{1 0 10021 7 1 0 100}", -       ! General Text, ISO Latin 1 and
 "{1 0 10021 7 1 0 1}", -         ! ISO 8859-1
 "{1 0 10021 7 1 0 77}", -        !
 "{1 0 10021 7 1 0 101}", -       ! ISO 8859-2
 "{1 0 10021 7 1 0 109}", -       ! ISO 8859-3
 "{1 0 10021 7 1 0 110}", -       ! ISO 8859-4
 "{1 0 10021 7 1 0 144}", -       ! ISO 8859-5
 "{1 0 10021 7 1 0 127}", -       ! ISO 8859-6
 "{1 0 10021 7 1 0 126}", -       ! ISO 8859-7
 "{1 0 10021 7 1 0 138}", -       ! ISO 8859-8
 "{1 0 10021 7 1 0 148}", -       ! ISO 8859-9
 "{1 0 10021 7 1 0 2}", -         ! General Text, IA5
 "{2 6 3 4 2}", -                 ! IA5text
 "{1 3 12 1011 1 3 1}", -         ! DDIF
 "{1 3 12 2 1011 5 5 1 0}")]      ! Any
```

11.5.4 Entering Content Information for the MAILbus 400 Message Router Gateway

The following is an example of the content information that you might choose to provide for individuals in a Message Router™ network who are served by the MAILbus 400 Message Router Gateway (XMR):

- Content Type

  This is 1992 Externally Defined IPMS, where any File Transfer bodypart is translated into an Externally Defined bodypart.

- Encoded Information Types (EITs)

  The EITs applicable to a Message Router network are:

  – DDIF

  – WPS-PLUS

  – IA5text

  – General Text, ISO Latin 1

  – General Text, IA5

  – DEC MCS

  – DECdx

    – Message Router Text
    – USA Nationally Defined
    – Any

  Note that DDIF must be the first encoded information type specified in the content information. This ensures that the MTA converts any bodypart types, such as ISO6937 or ODIF, which cannot be handled by the Gateway, to bodypart types that the Gateway can handle. You must include Any in the list of encoded information types.

- Content Length

  Whatever is applicable.

The following is an example of a directory entry for an individual served by the MAILbus 400 Message Router Gateway. Note that text following an exclamation mark (!) is not part of the command; it is added to explain the object identifiers in the command:

```
SET MTS "/MTS=ACME" ORADDRESS -
 "C=NZ; A=NZ-PTT; P=ACME; O=ACME; OU1=WELL; CN=John Smith" -
 CONTENT INFORMATION [MAXIMUM CONTENT LENGTH = 0, -
 CONTENT TYPES=( -
 "{1 3 12 2 1011 5 5 0 1 22}"), -   ! 1992 Externally Defined IPMS
 ENCODED INFORMATION TYPES = ( -    !
 "{1 3 12 1011 1 3 1 7}", -         ! DDIF
 "{1 3 12 1011 1 3 7}", -           ! WPS-PLUS
 "{2 6 3 4 2}", -                   ! IA5text
 "{1 0 10021 7 1 0 2}", -           ! General Text, IA5
 "{1 0 10021 7 1 0 1}", -           ! General Text, IA5
 "{1 3 12 2 1011 5 1 209}", -       ! DEC MCS
 "{1 0 10021 7 1 0 6}", -           ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 100}", -         ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 1}", -           ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 77}", -          ! General Text, ISO Latin 1
 "{1 3 12 1011 1 3 8}", -           ! DECdx
 "{1 3 12 2 1011 5 1 210}", -       ! Message Router Text
 "{1 3 12 2 1011 5 5 2 0 310}", -   ! USA Nationally Defined
 "{1 3 12 2 1011 5 5 1 0}")]        ! Any
```

11.5.5 Agents Using the Shared File 1984 Interface

This section describes the Retix® OpenServer 400™ cc:MAIL® Gateway to X.400. The instructions apply to all the PC LAN Gateways developed for the Retix OpenServer 400.

The OpenServer 400 cc:MAIL Gateway to X.400 is an example of a mail application that provides electronic mail services for a Local Area Network (LAN). This Gateway exchanges messages with the MTA using the Shared File 1984 interface.

Note that the OpenServer 400 cc:MAIL Gateway is the interface between the MTA and a cc:MAIL User Agent.

The following is an example of the Content Information that you might provide for individuals who are served by Agents using the Shared File 1984 interface:

- Content Type

  This is Interpersonal messaging 1984.

- Encoded Information Types (EITs)

  Recipients with these types of User Agent can only receive the following types of bodypart:

  - IA5text
  - Teletex
  - Bilaterally Defined

- Content Length

  Whatever is applicable.

The following is an example O/R address entry for an individual served by a Shared File 1984 Agent. Note that text following an exclamation mark (!) is not part of the command; it is added to explain the object identifiers in the command:

```
SET MTS "/MTS=ACME" ORADDRESS -
 "C=NZ; A=NZ-PTT; P=ACME; O=ACME; OU1=WELL: CN=Clare Curtis" -
 CONTENT INFORMATION [ MAXIMUM CONTENT LENGTH = 0, -
 CONTENT TYPES=( -
 "{1 3 12 2 1011 5 5 0 1 2}"),-   ! Interpersonal messaging 1984
 ENCODED INFORMATION TYPES = ( -  !
 "{2 6 3 4 2}", -                 ! IA5text
 "{2 6 3 4 5}", -                 ! Teletex
 "{1 3 12 2 1011 5 5 2 0 14}")]   ! Bilaterally Defined
```

11.5.6 Agents Using the Shared File 1992 Interface

This section provides an example of the Content Information that you need to specify for mail applications designed to be used with the ISOPLEX™ 800 MTA, for example ISOGATE™ for cc:Mail. This Access Unit exchanges messages with the MTA using the Shared File 1992 interface.

Read the documentation supplied with the Access Units to find out more information about the types of bodypart users can receive. The following is an example of the Content Information that you might provide:

- Content Type

  This is 1992 Externally Defined IPMS, where any File Transfer bodypart is translated into an Externally Defined bodypart.

- Encoded Information Types (EITs)
  - IA5text
  - Teletex
  - Bilaterally Defined

- Content Length

  Whatever is applicable.

The following is an example O/R address entry for an individual served by the ISOGATE Server and Access Units. Note that text following an exclamation mark (`!`) is not part of the command; it is added to explain the object identifiers in the command:

```
SET MTS "/MTS=ACME" ORADDRESS -
 "C=NZ; A=NZ-PTT; P=ACME; O=ACME; OU1=WELL: CN=Mari Nedd" -
 CONTENT INFORMATION [ MAXIMUM CONTENT LENGTH = 0, -
 CONTENT TYPES=( -
 "{1 3 12 2 1011 5 5 0 1 22}"),-  ! 1992 Externally Defined IPMS
 ENCODED INFORMATION TYPES = ( -  !
 "{2 6 3 4 2}", -                 ! IA5text
 "{2 6 3 4 5}", -                 ! Teletex
 "{1 3 12 2 1011 5 5 2 0 14}")]   ! Bilaterally Defined
```

Recipients served by the ISOCOR X.400 Router for Lotus Notes® can receive more types of bodypart. The following is an example of the directory entry for an individual served by the ISOCOR X.400 Router for Lotus Notes. Note that text following an exclamation mark (`!`) is not part of the command; it is added to explain the object identifiers in the command:

```
SET MTS "/MTS=ACME" ORADDRESS -
 "C=NZ; A=NZ-PTT; P=ACME; O=ACME; OU1=WELL: CN=Stan Holland" -
 CONTENT INFORMATION [ MAXIMUM CONTENT LENGTH = 0, -
 CONTENT TYPES=( -
 "{1 3 12 2 1011 5 5 0 1 22}"),-  ! 1992 Externally Defined IPMS
 ENCODED INFORMATION TYPES = ( -  !
 "{2 6 3 4 2}", -                 ! IA5text
 "{2 6 3 4 5}", -                 ! Teletex
 "{1 3 12 2 1011 5 5 1 1 11}", -  ! ISO 6937
 "{1 0 10021 7 1 0 6}",  -        ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 100}",  -      ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 1}",  -        ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 77}",  -       ! General Text, ISO Latin 1
 "{1 0 10021 7 1 0 102}", -       ! General Text, T.61
 "{1 0 10021 7 1 0 103}", -       ! General Text, T.61
 "{1 0 10021 7 1 0 106}", -       ! General Text, T.61
 "{1 0 10021 7 1 0 107}", -       ! General Text, T.61
 "{1 0 10021 7 1 0 2}", -         ! General Text, IA5
 "{1 0 10021 7 1 0 1}")]          ! General Text, IA5
```

11.5.7 Messages Transferred Across an MTS Based on the 1984 MHS Standards

This section describes what you need to consider if your routing domain connects to other routing domains based on the 1984 MHS Standards.

If you expect messages to be transferred across an intermediate routing domain based on the 1984 MHS Standards that cannot transfer the Interpersonal messaging 1992 content type, complete one of the following according to whether the recipient belongs to a MAILbus 400 MTA routing domain, or not:

- If the recipient belongs to a MAILbus 400 MTA routing domain, modify the recipient's Content Information in the directory such that the recipient is registered as being able to receive the Interpersonal messaging 1984 content type. You do not need to modify any of the preferred EITs, including those describing Externally Defined bodyparts.
- If the recipient belongs to a non-MAILbus 400 MTA routing domain, follow the recommendation described in Section 11.5.1.

12 Proprietary Content Information and IPM Bodypart Converters

This chapter describes how to assign object identifiers to proprietary content types and EITs that are used in your MHS. This chapter also describes the tasks that you must complete in order to set up the MTA to recognize IPM bodypart converters that you have written.

12.1 Assigning Object Identifiers to Proprietary Content Types, Data Formats or Bodyparts

Object identifiers uniquely identify an object. In the context of message transfer they are used, for example, to identify message content types and the EITs for data formats or bodyparts within these content types.

You may find that you are using proprietary content types, data formats or bodyparts in your routing domain which are not listed in Table 11–1 or Table 11–2. It is not possible for HP to supply object identifiers for proprietary content types, data formats or bodyparts. To be able to represent such formats in the content information of your O/R addresses, you need to assign object identifiers yourself, or use those already assigned by the responsible authority or by your organization.

If your organization has already been allocated object identifiers by a registration authority, then you can allocate object identifiers within your organization yourself. Ensure that the numbers you allocate are unique within your organization. Otherwise, contact a registration authority, such as the American National Standards Institute (ANSI) or the European Computer Manufacturers' Association (ECMA) to be allocated an object identifier for your organization.

An object identifier consists of a sequence of numbers enclosed in brackets, for example { 2 6 3 4 2 }. Each number has semantic significance, identifying an object and its position in a defined hierarchy. The first number in the sequence is either 0, 1 or 2, indicating that the object identified resides in the hierarchy under CCITT (0), ISO (1) or joint CCITT and ISO (2). For example, { 2 6 3 4 2 } is the joint CCITT and ISO object identifier for the IA5 text EIT.

ISO and CCITT are the registration authorities which allocate the second number in the sequence to another object, for example an organization. This organization then becomes the registration authority for subsequent numbers in the hierarchy. This system is similar to the system of allocating unique O/R address attributes. It makes it possible to order objects in a hierarchical scheme that is infinitely expandable and that enables the uniqueness of object identifiers on a global scale.

Using the example of the IA5 text EIT, the object identifier { 2 6 3 4 2 } is based on the following hierarchy:

Number	Identifies
2	A joint ISO CCITT object
6	A MOTIS-based MHS
3	An MTS in the MOTIS-based MHS
4	EITs (generally)
2	IA5 text

For more information about assigning object identifiers, see CCITT X.208 Recommendation, Section 28 and Annex D, as well as International Standard ISO 8824.

12.2 Proprietary Bodyparts and Converters

An MTA is able to transfer all the bodyparts described in Section 11.3, and can convert some of these bodyparts to other bodyparts. However, you may want to define your own bodypart or use a new bodypart that is not described by a Bodypart entity supplied with the MTA.

You may also want an MTA to do other conversions; for example, to convert other bodyparts to or from your own bodypart. An MTA can do this, provided you have written and installed the necessary converter.

The following sections explain how bodyparts are defined and how to integrate your converter with the MTA.

Note

This section does not explain how to write a converter; this is outside the scope of this guide.

Once you have integrated your converter with the MTA, it is possible to chain your converter to other converters to extend the range of conversions that the MTA can do on your bodypart, see Section 12.2.7.

12.2.1 How New Bodyparts are Recognized

An MTA is able to recognize a new proprietary, or a new standardized bodypart, provided that the bodypart is encoded as an Externally Defined bodypart type. Externally Defined bodyparts are defined in CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7, Section 7.3.12.

You might want to use a new bodypart, for example, if the MHS Standards are enhanced in the future or another organization has defined a bodypart. If you want an MTA to recognize a new bodypart not mentioned in Section 11.3, then create a Bodypart entity for the new bodypart definition, see Section 12.2.3.

12.2.2 How New Bodyparts are Defined and Encoded

All bodyparts are encoded using ASN.1 BER (Basic Encoding Rules) as defined by the CCITT in their Recommendations X.208 and X.209, and by ISO in their International Standards ISO 8824 and ISO 8825.

X.420 and ISO/IEC 10021-7 provide two definitions of their bodypart types, a **basic bodypart** definition and an **extended bodypart** definition. The bodypart definition determines how the bodypart is encoded. The basic definition is used for backwards compatibility with the 1984 MHS Standards. There is a basic definition for all bodyparts defined in the 1984 MHS Standards.

Use the extended bodypart definition for an Externally Defined bodypart to encode a bodypart that you define. The ASN.1 of an extended bodypart starts with the context-specific tag `[15]`. The bodypart can consist of two components, a parameters component, which is optional, and a data component. The parameters component provides accompanying information for the type of data contained in the data component. The data component contains the data. Both components are identified by an object identifier that you must assign.

Section 12.1 explains how to obtain object identifiers for your Externally Defined bodypart. Also, for advice about defining your own Externally Defined bodypart, see Part 8, Annex B, of the OIW Stable Implementation Agreements. For details of how to obtain this document, see Appendix A.

Note

Do not use the same object identifier to describe the data components of different bodypart types.

12.2.3 Creating a Bodypart Entity

So that an MTA can recognize a new bodypart you need to create a Bodypart entity at the MTA. A Bodypart entity specifies the mapping between the Encoded Information Types (EITs) that describe the contents of the bodypart and the identifier for that bodypart.

Use the following command to create a Bodypart entity:

```
CREATE NODE "node-id" MTA BODYPART "name" -
ENCODED INFORMATION TYPES {"{eit}", "{eit}"}, -
IDENTIFIER "{id}"
```

where:

- *name* is the name of the bodypart.

  You can use any name, however, the name must be unique within your routing domain and consist only of alphanumeric characters.
- *eit* is a set of object identifiers that specify the bodypart's encoded information types.
- *id* is the object identifier that identifies the bodypart.

  In the Externally Defined bodypart encoding, this is the object identifier that describes the bodypart's data component.

You can use the same object identifier that identifies the data component as the value for the Encoded Information Types argument, for example:

```
CREATE NODE "node-id" MTA BODYPART "myformat" ENCODED INFORMATION TYPES -
{"{1 2 3 4 5}"}, IDENTIFIER "{1 2 3 4 5}"
```

Using the same object identifier for both arguments ensures that the bodypart is unique and gives a one-to-one mapping between the EIT and the identifier.

Create a Bodypart entity at each MTA in your routing domain, not just at the MTA that does the conversion. This ensures that all the MTAs in your routing domain recognize the bodypart.

12.2.4 Converter Requirements

You can write a converter to convert one bodypart to another. Your converter can convert between X.400-defined bodyparts, or proprietary bodyparts, or a mixture of both. This section describes the interface between the MTA and a converter and how to integrate a proprietary converter with the MTA. Each converter is an independent image invoked as a separate process from the MTA. For a converter to work with the MTA, it must do the following:

- Provide an input and output mechanism for data.

Tru64 UNIX

On Tru64 UNIX, the converter reads bodypart data from its standard input stream “stdin”. The converter sends data to the MTA through its standard output stream “stdout”. If your converter generates error messages, then error messages are sent to standard error, that is, “stderr”.

♦

OpenVMS

For a converter implemented as an OpenVMS image (.EXE file), the converter reads bodypart data from its standard input stream SYS$INPUT. The converter sends data to the MTA through its standard output stream SYS$OUTPUT. If your converter generates error messages, then error messages are sent to standard error, that is, SYS$ERROR.

For a converter implemented as an OpenVMS command procedure (.COM file), the MTA invokes the converter with two arguments: the first argument is the input file and the second argument is the output file. The standard error SYS$ERROR is not available for command procedures.

♦

- Return a specific value when the conversion is successful.

Tru64 UNIX

On Tru64 UNIX, when a converter successfully converts a source bodypart, it returns a 0 (zero) exit status to the MTA. This indicates to the MTA that the conversion was successful. If the converter is unsuccessful, the converter must return to the MTA an exit status of 1.

♦

OpenVMS

On OpenVMS, when a converter successfully converts a source bodypart, the exit status returned to the MTA must have the least significant bit set to 1. This indicates to the MTA that the conversion was successful. If the conversion is unsuccessful, then the converter returns to the MTA an exit status which has the least significant bit set to 0 (zero). These exit values should be returned by the VMS SYS$EXIT routine. It is recommended that you do not use the C runtime library exit( ) routine.
♦

- Accept a bodypart encoded in ASN.1 (BER), as defined by CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7.

  The MTA always supplies a source bodypart encoded in ASN.1 (BER) to a converter.

- Generate a bodypart encoded in ASN.1 (BER), as defined by CCITT Recommendation X.420 and International Standard ISO/IEC 10021-7.

  The MTA expects to receive a target bodypart encoded in ASN.1 (BER) from a converter.

- Accept the context-specific tag of a bodypart.

  An MTA always supplies source bodyparts with their leading context-specific tag to a converter; for example, an IA5 bodypart has the leading context-specific tag `[0]`. The MTA also expects to receive a bodypart that starts with a leading context-specific tag from a converter. For your own proprietary bodyparts, you would use the context-specific tag `[15]` introducing the Externally Defined bodypart definition.

- Accept an X.400-defined bodypart that is encoded as a basic bodypart.

  When there is a choice of bodypart encoding, as defined in X.420 and ISO/IEC 10021-7, the MTA always supplies a basic bodypart. As an example, an IA5 bodypart can be encoded as either a basic bodypart or as an extended bodypart.

  Note that there is one exception to this rule - ODA bodyparts are always encoded as extended bodyparts.

- Generate an X.400-defined bodypart encoded as a basic bodypart when appropriate.

  When there is a choice of bodypart encoding, as defined in X.420 and ISO/IEC 10021-7, then your converter must output a basic bodypart. This ensures that your converter can be used in a sequence of converters, see Section 12.2.7. As an example, an IA5 bodypart can be encoded as either a basic bodypart or as an extended bodypart.

Note that there is one exception to this rule - ODA bodyparts are always encoded as extended bodyparts.

12.2.5 Integrating a Converter with an MTA

To integrate your converter with the MTA do the following:

1. Install your converter image at the MTA that is to do the conversion, for example the MTA where the Agent requiring the conversion is located.

   Converter images are held in the MTA's converter image directory. Your converter image should have the same file ownership and file protection as the converters that are supplied with the MTA.

   For the location of the MTA's converter image directory and the applicable ownership and protection values, refer to the appendix describing the operating system specific information.
2. Create Bodypart entities, if they do not already exist, that describe the bodyparts that are referenced by your converter at each MTA in your routing domain, see Section 12.2.3.
3. Create a Converter entity that describes the converter image at the MTA where you have installed your converter, see Section 12.2.6.
4. Include the commands that create your Converter and Bodypart entities in the MTA's startup script.

Note

You must create the Bodypart entities referenced by your converter before you can create the Converter entity that describes your converter.

12.2.6 Creating a Converter Entity that Describes a Converter Image

At the MTA where you have installed your converter, create a Converter entity that describes your converter image.

Note

The name of the Converter entity must match the name of the converter image. On OpenVMS systems, the name of a converter image has the file extension .EXE, for example, IA5TOLATIN1.EXE. Do not include the .EXE extension in the name of the Converter entity.

Use the following command to create a Converter entity:

```
CREATE NODE "node-id" MTA CONVERTER "name", SOURCE "source", -
TARGET "target", STEPS (steps), LOSSY lossy
```

where:

- `name` is the name of the Converter image.

  You can use any name, but the name must consist only of alphanumeric characters and identify an image file in the MTA's converter image directory. For the location of the MTA's converter image directory, refer to the appendix describing the operating system specific information.
- `source` is the name of the Bodypart entity that describes the input bodypart to the converter.
- `target` is the name of the Bodypart entity that describes the output bodypart from the converter.
- `steps` is a list of converter names that defines a sequence of conversions; this value must be empty when you create a Converter entity that describes a converter image.
- `lossy` specifies whether data loss during conversion is possible, its value is either `TRUE` or `FALSE`.

12.2.7 Creating a Converter Entity that Specifies a Sequence of Converters

You can include your converter in a sequence of converters. You do this by creating a Converter entity that specifies the names of the converters to be called in sequence in the Steps argument. The following rules apply when you specify a sequence of converters:

- The input bodypart to the sequence must be different from the output bodypart from the sequence.
- The input bodypart to the sequence must correspond to the Bodypart entity specified in the Source attribute of the first converter in the sequence.
- The output bodypart from the sequence must correspond to the Bodypart entity specified as the Target attribute of the last converter in the sequence.
- Each converter named in the sequence must have a corresponding Converter entity.
- Each Converter entity named in the sequence must describe a converter image that converts directly from one bodypart to another.

For example, supposing you are using a proprietary bodypart called “myformat”. You have also written and installed a converter called “myformattolatin1”, which converts “myformat” to ISO Latin 1 characters. You could run your converter in sequence with any of the MTA’s converters that take ISO Latin 1 as their input, for example, “latin1tot61”. To do this you need to create a Converter entity that specifies the appropriate sequence of conversions in its Steps argument. You must create this Converter entity at the MTA where your converter image, and the other converter images in the sequence, are installed. Use the following command to create a Converter entity:

```
CREATE NODE "node-id" -
    MTA CONVERTER "myformattot61",  -
    SOURCE "myformat", -
    TARGET "teletex", -
    STEPS ("myformattolatin1", "latin1tot61"), -
    LOSSY = TRUE
```

Once you have created this Converter entity, the MTA is able to convert bodyparts containing “myformat” data to bodyparts containing teletex data.

13
Event Dispatching

This chapter describes where events issued by entities of the MTA module are dispatched and gives information about how to tune MTA event dispatching to suit your management requirements.

13.1 The Event Dispatching Mechanism

The MTA entity and some of its subentities post events whenever something noteworthy happens. While some MTA events are informational, most MTA events indicate that an error has occurred. See the *MTA Module Online Help* or Part III for a description of the events that the MTA issues. Note that the entities of the MTS module do not issue events.

To distribute events, the MTA uses the HP DECnet-Plus event dispatching mechanism. All events issued by HP DECnet-Plus applications, including the MTA, are dispatched to the HP DECnet-Plus Event Dispatcher module. The entities of the Event Dispatcher module are responsible for dispatching events to event sinks that you can specify. The Event Dispatcher module and the event dispatching mechanism for all HP DECnet-Plus applications are described in the HP DECnet-Plus documentation. This section on MTA events complements, but does not repeat, the information in these documents.

Note

You tune dispatching of MTA events through the entities of the Event Dispatcher. You cannot use MTA management to tune the dispatching of MTA events.

Tru64 UNIX

On Tru64 UNIX, each MTA has its own NCL script that enables MTA event dispatching. For the name and location of this script, refer to the appendix describing the operating system specific information. Unless modified, this script creates and enables an outbound event stream for all MTA events and issues these to all sinks on the local system. You can change the default settings; Section 13.2 shows possible ways of setting up MTA event dispatching to suit your specific management requirements.

During installation of an MTA, the script to start MTA event dispatching is automatically executed. To ensure that this script is executed at a system reboot, a command is automatically added to the OSI applications startup file which executes the script to start MTA event dispatching.

♦

OpenVMS

On OpenVMS, events issued by most applications, including the MTA, are automatically passed by the Event Dispatcher to the PhaseV_Sink event sink on the local system, unless you have modified event dispatching on your system.

The MTA provides an NCL script that you can use to create and enable an outbound stream and event sink specifically for MTA events. The commands in the MTA's event dispatching script are commented out; should you want to use these commands, remove the comment (!) from each command line. For the name and location of the MTA's event dispatching script see Appendix E. You can modify the commands in this script; see Section 13.2 for possible ways of setting up MTA event dispatching to suit your specific management requirements.

If you are using the MTA's event dispatching script, ensure that this script is automatically executed at a system reboot. To do so, check that you have added the command which executes the MTA's server initialization procedure to the system startup file (as described in Part III of *HP MAILbus 400 MTA Planning and Setup*).

♦

For more detailed information on how to view events see *MAILbus 400 Getting Started*.

The MTA's event dispatching script is not automatically executed if the Event Dispatcher or DECnet is stopped and restarted. Therefore, if you stop and restart the Event Dispatcher or DECnet, you must re-enable MTA event dispatching by manually executing this script. Refer to the appendix describing the operating system specific information for the command to use.

13.2 How to Tune Dispatching of MTA Events

You can tune the dispatching of MTA events in the following ways:

- Specify event sinks that are different from the local event sink.

  This depends on your method of network management:

  - Centralized network management

    If you manage your network and the MTAs in your routing domain from one central node, you can have events from all MTAs in your routing domain dispatched to the event sinks located on that central node. Alternatively, you can create a new event sink on the central node exclusively for MTA events.

  - Decentralized network management

    If you have decentralized network management by dividing your network into management regions, you may have an event sink in each region. Then, events from all MTAs in a management region could be dispatched to one selected event sink. Alternatively, you can create new event sinks exclusively for MTA events.

  The script to start MTA event dispatching contains a command that is commented out and that redirects the outbound event stream to another node. If you intend to redirect MTA events to the event sinks on a particular remote node, supply the name of the node in the applicable command and remove the comment. MTA events will then be dispatched to the event sinks on the node that you have specified.

  This script also contains a section that is commented out and that contains commands that create an MTA-specific event sink. If you want to create an event sink on a node specifically for MTA events, supply the required information in the commands, as indicated in the script, and remove the comments from the section.

- Prevent certain MTA events from being dispatched.

  Information about how to block individual events is provided in the HP DECnet-Plus documentation.

- Dispatch specific MTA events to specific event sinks.

You may want to do this, for example, if you have divided network management by task rather than by region. You could then report one type of event to one event sink, for example, events indicating problems with the directory, while system-related events such as disk space problems could be reported to another.

Information about how to redirect specific events to specific event sinks is provided in the appendix describing the operating system specific information.

- Add events from the OSI Transport module.

  The events issued by the entities of the OSI Transport module are of interest to MTA management. They indicate problems that affect MTA operation and that cannot be reported by the MTA itself.

  If you have set up an event sink specifically for MTA events, you can have events from the OSI Transport module reported to the MTA event sink. Note that in this case, you may receive multiple copies of OSI Transport events, in the MTA event sink and in other event sinks you may be using.

  If you monitor events issued by a group of MTAs from a centralized, remote event sink, you can have events issued by the OSI Transport module from the MTA nodes reported to this centralized sink. You do this by setting up the MTA event stream to include OSI Transport events.

  Information about the OSI Transport module can be accessed as a topic in the *NCL Online Help*. Information about how to add events to event streams and event sinks is provided in the HP DECnet-Plus documentation.

In order to permanently modify the way an MTA's events are dispatched, you must enter your changes in the script to start MTA event dispatching. This ensures that these changes take effect whenever this script is executed.

Part III

Solving Problems

This part explains how you can solve message transfer problems within your routing domain. This part contains the following chapters:

- Chapter 14, which gives an overview of the problems that can occur in your routing domain and your scope for solving them.
- Chapter 15, which describes the circumstances when you can receive the Access Denied error when modifying entries in the directory.
- Chapter 16, which explains how to solve problems with associations and the protocols of the OSI protocol stack on which the X.400 messaging protocols operate, that is, RTSE, ACSE, Presentation Layer and Session Layer. The chapter also explains how to record protocol information so that you can trace protocol errors.
- Chapter 17, which explains the MTA's use of OSI Transport module and X.25 modules when making inbound and outbound connections.
- Chapter 18, which explains how to solve problems with messages. The chapter also describes how to trace a message in your routing domain.
- Chapter 19, which explains how to solve problems with the routing of messages.
- Chapter 20, which explains how to solve problems with the resources available to an MTA.
- Chapter 21, which explains how to solve problems related to receiving events, collecting Accounting information, Archiving, and Message History logging.
- Chapter 22, which explains the events that report failure of the MTA.
- Chapter 23, which explains how to report problems that you cannot solve to HP.

14 Overview of MTS Problems

This chapter defines the scope of MTS problem solving and describes how you determine that a problem exists. This chapter helps you recognize the type of problem and tells you where to look for the information you need to solve it.

14.1 Your Scope for Solving MTS Problems

The problems you can solve within an MTS are determined by:

- The topography of the MTS

  An MTS can be divided into a number of routing domains. If you have the required privileges you can manage any MAILbus 400 MTA from any node on the network that has MAILbus 400 MTA management installed. However, problem solving is applicable only within HP routing domain boundaries (see Figure 14–1).

- The scope of your management responsibilities

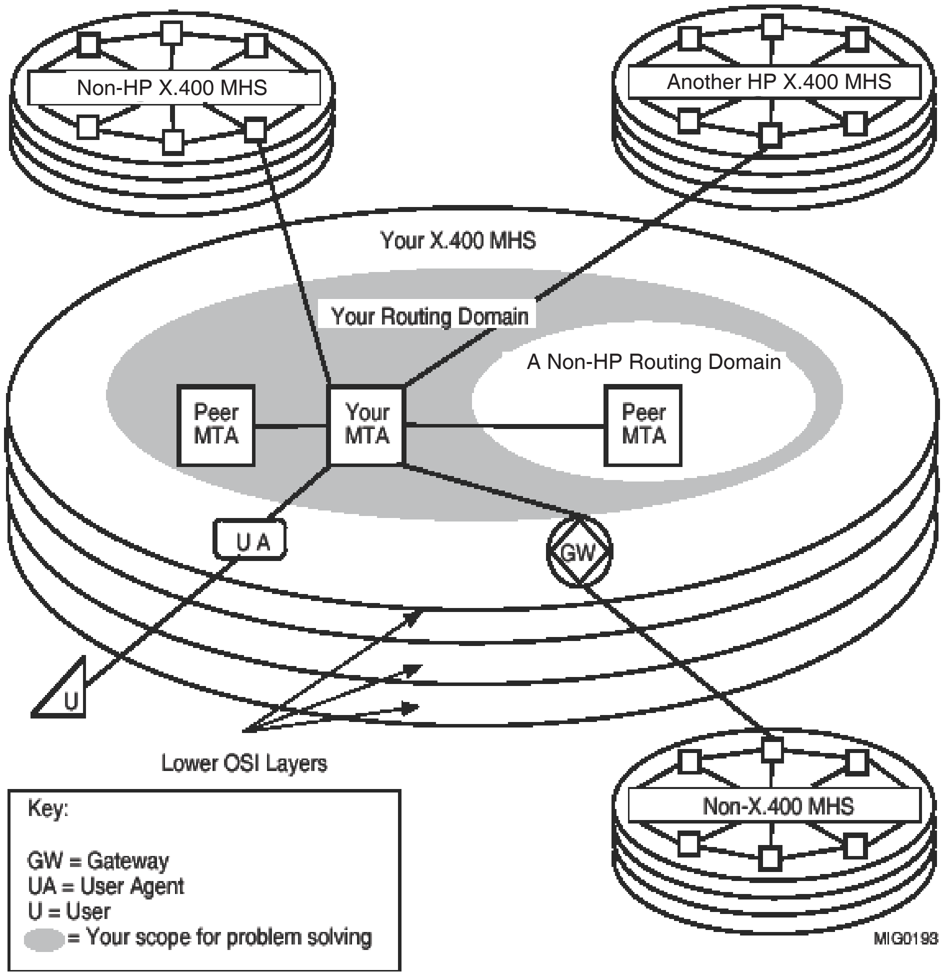
  A routing domain can be divided into a number of regions for management purposes. You can solve problems within any routing domain for which you have management responsibility for all the MTAs. You can also investigate problems at the routing domain's boundaries.

- How individual MTAs within the MTS are configured

  The scope for problem solving can be limited because one or more MTAs within a routing domain have been configured in a way that does not provide the necessary data. For example, if Message History logging has not been enabled or has failed at an MTA it is not possible to trace the path of a message through your routing domain. See Section 18.4 for information about tracing messages.

Your scope for problem solving is illustrated in Figure 14–1. You cannot solve problems directly if they originate in a different routing domain or in a non-HP routing domain.

Figure 14–1 Your Scope for Problem Solving



If a problem is outside your scope for problem solving, you can follow it up according to where it originates, as follows:

- Another routing domain

  Make sure you are familiar with the characteristics of any relevant boundary MTA or Gateway in your own routing domain. Then communicate with the manager responsible for the other routing domain.

- A peer MTA in another routing domain or management region

  Make sure you are familiar with the characteristics of any relevant MTA in your routing domain or management region. Check for special bilateral arrangements with any peer MTA which may be involved. Then communicate with the manager responsible for the peer MTA.

  To make it easier to contact managers in other routing domains, set up the Contact Name attribute of each Peer MTA entity that you create. Use this attribute to hold the name and telephone number of the person responsible for managing the peer MTA represented by the Peer MTA entity.

  Use the following command to set this attribute:

  ```
  SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
  NAME = "peer-mta-name"] CONTACT NAME = "text"
  ```

  where *peer-mta-name* is the name of the Peer MTA entity and *text* is information that identifies the manager of the peer MTA.

- An Agent served by an MTA for which you are responsible.

  Collect any information about the problem that you can. Ask the person responsible for the Agent to investigate.

- A product from another vendor or a lower OSI layer.

  Consult the relevant documentation or the manager of the product or lower OSI layer and, if necessary, the product vendor.

14.2 How You Become Aware of Problems

You become aware of problems in two ways:

- From MHS users

  A user who tells you of a problem may be a message originator or a recipient. The user may have received a non-delivery report or noticed something else that was unsatisfactory. For example, an originator could discover that a message was not correctly transferred because the intended recipient did not respond as expected. For further information about the problems likely to be noticed by MHS users, see Section 14.2.1.

- From events and changes in counter attributes

  The MTA generates events that are sent to one or more event sinks. Each occurrence of an event is recorded by a counter attribute, see Section 14.2.2.

14.2.1 Problems Notified by MHS Users

The types of problems that an MHS user might bring to your notice are shown in Table 14–1.

Table 14–1 Problems Notified by MHS Users

Problem	Probable Causes
Multiple copies of a message received	An MTA on the message's route failed (for example, because of an operating system failure which caused the MTA to stop) and the MTA recovered while that message was being transferred (see Section 5.1).
Message not delivered	There were problems with the message itself, with routing or with processing. If a message was not received as expected, and/or if a non-delivery report was not received, investigate the reasons by tracing the message (see Section 18.4).
Message content corrupted	A message contains invalid characters. For example, '?' appears as '&'. A bodypart in the message had to be converted to make it acceptable to the recipient's Agent. The conversion took place, but the message contains text that is still unacceptable to the recipient's Agent. This error is due to a problem with either the data supplied to the converter or to the converter failing to do the conversion correctly.

(continued on next page)

Table 14–1 (Cont.) Problems Notified by MHS Users

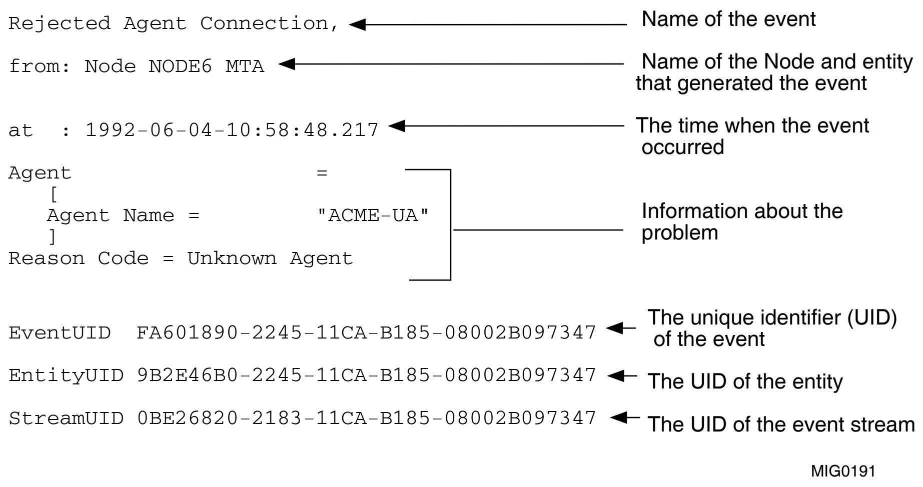
Problem	Probable Causes
Distribution list not expanded	An MTA on the message's route has had problems accessing the directory, or the distribution list was unavailable or incorrect. Reasons can be found by tracing the message (see Section 18.4) or by using the ORaddress entity of the MTS module to examine the distribution list entry in the directory.
Expected redirection failed	An MTA on the message's route had problems accessing the directory entry concerned with redirection or the redirection instruction is incorrect. Reasons can be found by tracing the message (see Section 18.4) or by using the ORaddress entity of the MTS module to examine the redirection information in the recipient's O/R address held in the directory.
Probe failed	The failure is due to the reason reported to the originator, or characteristics of the probe were unacceptable to the recipient. Use the ORaddress entity to check the recipient's O/R address information in the directory. In particular, check the EIT and content length information in the recipient's O/R address.
Delivery or Non-delivery report not received	Reporting was disabled on originator's instructions, or a report was redirected or discarded by an MTA on its route. If a report was not received as expected and not redirected or discarded, reasons should be investigated. You can use MPDU entities to trace reports still held in an MTA (see Section 18.6).

In some circumstances, users may perceive message transfer problems although the MTAs in your routing domain are functioning normally. A likely cause of such problems is the specific way that User Agents are being used; for example, a User Agent has been set up so that is does not deliver reports, such as non-delivery reports, to users. Another likely cause is inconsistencies in the information that the MTS requires for routing, see Chapter 19.

14.2.2 Problems Revealed by Events and Counter Attributes

Events are generated and sent to event sinks when a particular problem occurs in your routing domain or when certain entities are deleted by a management command. Events, or combinations of events, indicate particular types of problem, and also indicate MTA state changes.

The following is an example of a typical event:



```
Rejected Agent Connection,

from: Node NODE6 MTA

at  : 1992-06-04-10:58:48.217

Agent                  =
   [
   Agent Name =         "ACME-UA"
   ]
Reason Code = Unknown Agent


EventUID  FA601890-2245-11CA-B185-08002B097347

EntityUID 9B2E46B0-2245-11CA-B185-08002B097347

StreamUID 0BE26820-2183-11CA-B185-08002B097347
```

Name of the event
Name of the Node and entity that generated the event
The time when the event occurred
Information about the problem
The unique identifier (UID) of the event
The UID of the entity
The UID of the event stream
MIG0191

See Table 14–2 for the reference section that describes each particular event.

Note that problems are likely to show up as a sequence of related events. For example, you receive an MPDU Expired event, this event would have been preceded by an Expiry Alarm Threshold Exceeded event. The MPDU could have been delayed in the MTA because the MTA was unable to establish an outbound association to the MPDU's target peer MTA. When the establishment of the association failed, the MTA generated the Outbound Establishment Failure event. If the abort was caused by a protocol error, then the MTA also generates the RTSE Protocol Violation event or the Lower Layer Protocol Violation event.

Because of the relationship between events associated with specific problems, it is important to check the timestamp associated with any events under investigation. Note that during a forced exit by the MTA, an exceptional situation occurs in which timestamps can be misleading (see Chapter 22).

Apart from the Forced Exit and Entity Deleted events, each occurrence of an event is recorded by a corresponding counter. For example, occurrences of the MPDU Expired event are recorded by the Expired MPDUs counter of the MTA entity.

This enables you to keep a quantitative check on the frequency of each type of event for which there is a counter. Some events, such as the Inbound Transfer Hard Rejection event, are sufficiently serious to require investigation of each occurrence. Other events, such as the Expiry Alarm Threshold Exceeded event, indicate problems only if they occur frequently. Counters are, therefore, useful for checking the frequency of events, but you have to monitor the levels of counters regularly if you want to use them to help in problem solving.

Use the Show command in NCL to display the value of a counter. You can display all the counters of a particular entity by including the `ALL COUNTERS` qualifier in the Show command. See the *MTA Module Online Help* for information about the `ALL COUNTERS` qualifier.

An entity's Creation Time attribute shows when the counters were set to zero. Use this attribute as a reference point to when the counters started recording information.

Table 14–2 lists each event, its related counter, and the entity that generates the event.

Table 14–2 Events and Related Counter Attributes

Event	Related Counter	Reference
MTA Entity		
Accounting Data Lost	Accounting Data Losses	Section 21.2.1
Converter Unavailable	Unavailable Converters	Section 18.7.8
Deferred Message Deleted	Deleted Deferred Messages	Section 18.7.6
Directory Configuration Error	Directory Configuration Errors	Section 19.3.4
Directory Service Error	Directory Service Errors	Section 20.2
Expiry Alarm Threshold Exceeded	Expiry Alarms	Section 18.7.1
Forced Exit	not counted	Section 22.2.2
Inbound Transfer Hard Rejection	Inbound Transfer Hard Rejections	Section 16.2.3
Inbound Transfer Soft Rejection	Inbound Transfer Soft Rejections	Section 16.2.5

(continued on next page)

Table 14–2 (Cont.) Events and Related Counter Attributes

Event	Related Counter	Reference
MTA Entity		
Internal Error	Internal Errors	Section 22.2.1
Invalid MPDU Detected	Invalid MPDUs Detected	Section 18.7.2
Licensed Message Throughput Exceeded	Licensed Message Throughput Exceeded	Section 22.3
Loop Detected	Loops Detected	Section 19.3.5
Message History Data Lost	Message History Data Losses	Section 21.2.2
MPDU Deleted	Deleted MPDUs	Section 18.7.7
MPDU Expired	Expired MPDUs	Section 18.7.3
Recovery Finished	not counted	Section 18.8.4
Rejected Agent Connection	Rejected Agent Connections	Section 19.3.1
Report Discarded	Reports Discarded	Section 18.7.4
Report Generation Failed	Report Generation Failures	Section 18.7.5
State Change	State Changes	Section 14.3
System Interface Error	System Interface Errors	Section 20.1
Transport Interface Error	Transport Interface Errors	Section 16.2.8
Unknown Agent	Unknown Agents	Section 19.3.2
Unknown Peer Domain	Unknown Peer Domains	Section 19.3.3
Agent Entity		
Archive Failed	Failed Archives	Section 21.2.3
Peer MTA Entity		
Archive Failed	Failed Archives	Section 21.2.3
Entity Deleted	not counted	Section 7.2.1

(continued on next page)

Table 14–2 (Cont.) Events and Related Counter Attributes

Event	Related Counter	Reference
Activity Entity		
Inbound Failure	Inbound Failures[1]	Section 16.2.6
Lower Layer Protocol Violation	Lower Layer Protocol Violations[1]	Section 16.3.2
Outbound Establishment Failure	Outbound Establishment Failures[1]	Section 16.2.1
Outbound Failure	Outbound Failures[1]	Section 16.2.7
Outbound Hard Rejection	Outbound Hard Rejections[1]	Section 16.2.2
Outbound Soft Rejection	Outbound Soft Rejections[1]	Section 16.2.4
RTSE Protocol Violation	RTSE Protocol Violations[1]	Section 16.3.1

[1]Peer MTA entity counter

Not all counters are related to events; for example, the MTA, Agent, and Peer MTA entities have counters that record the number of MPDUs that are handled by an MTA. Table 14–3 lists the counters that record statistical information not related to events.

Table 14–3 Counters not Related to Events

Counter	Entity
Delivered MPDUs	MTA
Exported MPDUs	MTA
Imported MPDUs	MTA
Inbound Acceptances	Peer MTA
Inbound Disconnections	Peer MTA
MPDUs In	Agent and Peer MTA
MPDUs Out	Agent and Peer MTA
Octets In	Peer MTA
Octets Out	Peer MTA

(continued on next page)

Table 14–3 (Cont.) Counters not Related to Events

Counter	Entity
Outbound Acceptances	Peer MTA
Outbound Disconnections	Peer MTA
Submitted MPDUs	MTA

When a Peer MTA entity is deleted, it generates the Entity Deleted event. This event contains the final values of the counters of the deleted Peer MTA entity. Section 7.2.1 describes how the information contained in this event can be useful if you are monitoring the message traffic in your routing domain.

14.3 MTA Changes in State

The State Change event is generated whenever the MTA changes state, for example, when you issue the DISABLE MTA NCL command.

This event is counted by the MTA entity's State Changes counter.

The event provides the following information:

```
Previous State= previous-state.
Current State= current-state.
```

where *previous-state* is the MTA state before the State Change was issued, and *current-state* is the new state of the MTA.

Action

This event is for your information, you do not need to take any action as a result of receiving this event.

15
Problems Accessing Routing Information

This chapter describes problems and errors that you might see when accessing routing information in the directory.

15.1 The MTS Entity and the Access Denied Error

If you receive the error Access Denied when attempting to modify entries in the directory, check the following:

- That you have the correct privileges.

  You must issue commands that modify routing information from a privileged account.

- That you have specified a password when creating the routing domain entry in the directory using the MTS entity.

  If you using the Create MTS command to create the routing domain entry for your routing domain in the directory for the first time, follow the instructions in Part III of *HP MAILbus 400 MTA Planning and Setup*. Read the chapter appropriate to your operating system describing how to set up the MTA.

- That the node is authorized to manage the specified routing information.

  Check that the node where you are issuing the MTS module commands is authorized to access the MTA's routing information. The password protecting the routing information might have been modified in the directory, or the node might not be authorized to manage the routing information for this MTA. In either case, find out the correct password to quote with the MTS entity at the node, as it has been modified. If you cannot find the correct password to use with the MTS entity read Section 15.3.

  The following is an example of the NCL command to use to provide the necessary authorization to manage the MTA's routing information from the node. Issue this command on the node that you want to authorize:

  ```
  AUTHORIZE MTS "mts_name" PASSWORD "password"
  ```

where:

- `mts_name` is the identifier of the MTS entity.
- `password` is the password for the MTS entity.

- That the DSA configuration is correct.

  If you have set up the MTA to contact a shadow DSA, there might be a problem with "Trust" between the DSAs. Refer to *HP Enterprise Directory - Problem Solving* for information about how to solve problems concerned with lack of "Trust" between DSAs.

15.2 MTS Dump Command Failures

You also receive the Access Denied error if you specify an incorrect password with the Dump command.

If the Dump command returns the error Access Denied, you are either not quoting a password when one is required, or you are quoting the incorrect password.

Contact the person responsible for managing the routing information and find out the correct password for your routing information, that is, the MTS entity password. If you cannot find out the correct password, refer to Section 15.3.

The following is an example of how to use the Dump command to dump the MTA's routing information to a named file:

```
DUMP MTS mts_name FILENAME filename, PASSWORD password
```

where:

- `mts_name` is the identifier of the MTS entity.
- `filename` is the full pathname (Tru64 UNIX) or file specification (OpenVMS) where you want to dump your routing information.
- `password` is the password for the MTS entity.

HP MAILbus 400 MTA Planning and Setup provides information about the MTS entity and directory authorization. The *MTS Module Online Help* provides more details about how to use the Dump command.

15.3 Defining a New Password

If you have forgotten the password for the MTS entity and need to set a new one, complete the following steps:

1. Temporarily create a DSA Accessor entity for the routing domain entry at the DSA that the MTA contacts as follows:

   ```
   CREATE NODE "node-id" DSA  ACCESSOR "mts_name" PASSWORD "temp"
   ```

   where *mts_name* is the identifier of the MTS entity.

2. Set a new password for the MTS entity as follows:

   ```
   SET MTS "mts_name" EXISTING PASSWORD "temp", PASSWORD "rememberthistime"
   ```

   where *mts_name* is the identifier of the MTS entity.

3. Delete the DSA Accessor entity:

   ```
   DELETE NODE "node-id" DSA ACCESSOR "mts_name"
   ```

   where *mts_name* is the identifier of the MTS entity.

16 Problems with Associations

This chapter describes problems that can occur with associations between MTAs. Table 16–1 shows how failure at each stage of an association relates to the events generated.

Note

Within this chapter, the term "association" refers to X.400 1988 OSI associations; however, the term also refers to X.400 1984 OSI Session connections when applicable.

Table 16–1 Problems at Different Stages of an Association

Initiation of Connection	Negotiation of Validity or Acceptability	Negotiation of Availability	Data Transfer Readiness	Event Generated
Fail	—	—	—	Outbound Establishment Failure (see Section 16.2.1) or Transport Interface Error (see Section 16.2.8) and either RTSE Protocol Violation (see Section 16.3.1) or Lower Layer Protocol Violation (see Section 16.3.2).
Pass	Fail	—	—	Outbound Hard Rejection (see Section 16.2.2) or Inbound Transfer Hard Rejection (see Section 16.2.3)
Pass	Pass	Fail	—	Outbound Soft Rejection (see Section 16.2.4) or Inbound Transfer Soft Rejection (see Section 16.2.5)

(continued on next page)

Table 16–1 (Cont.) Problems at Different Stages of an Association

Initiation of Connection	Negotiation of Validity or Acceptability	Negotiation of Availability	Data Transfer Readiness	Event Generated
Pass	Pass	Pass	Fail	Outbound Failure (see Section 16.2.7) or Inbound Failure (see Section 16.2.6) and either RTSE Protocol Violation (see Section 16.3.1) or Lower Layer Protocol Violation (see Section 16.3.2)

You can find out about problems with associations or with the establishment of an association by doing one of the following:

- Monitor an association using the Activity entity that provides information about the association (Section 16.1).
- Monitoring events that relate to association failures, establishment failures, or rejected attempts to establish an association (Section 16.2).
- Monitoring events that relate to protocol violations (Section 16.3).
- Monitoring the counters that record occurrences of the events described in Section 16.2 and Section 16.3.

Some association failures are due to errors in the OSI protocols that are exchanged between MTAs that are attempting to establish an association. OSI protocol errors are reported by the appropriate event. Usually, to solve this type of problem, you need to record the protocol information and trace the error, see Section 16.4.

16.1 Using the Activity Entity to Find Out When an Association Fails

The Activity entity provides information about an association and the transfer of an MPDU over that association. An Activity entity is created when the first MPDU is transferred over the association. During the establishment phase of an association, before MPDU transfer has started, no Activity entity exists. Therefore, you can only use the Activity entity to monitor an association after it has been established and MPDU transfer has started.

The Activity entity holds information about the association in its State and Interruption Reason attributes. The State attribute of an Activity entity refers to the current state of the association. The State attribute can have one of the following states:

- Active

  An MPDU is being transferred over the association.
- Idle

  A complete MPDU has been transferred, and the association is waiting for another MPDU to transfer.
- Interrupted

  The association has failed.
- Establishing

  The association is in the process of being established.

When an association fails, the Interruption Reason attribute provides information about the cause of the failure. For example, an MTA detects a protocol error and aborts an association. The Activity entity that holds information about the association has the following values in its State and Interruption Reason attributes:

```
State = Interrupted

Interruption Reason =
      [
      RTSE Entity that Initiated Failure = This MTA
      Abort Reason Code = Protocol Violation
      Local Diagnostic =  Not Applicable
      ]
```

Use the following command to display the State and Interruption Reason attributes of all the current Activity entities for all the associations to or from a particular peer MTA:

```
SHOW NODE "node-id" MTA PEER MTA identifier ACTIVITY * INTERRUPTION REASON, -
WITH STATE = INTERRUPTED
```

where `identifier` is either:

– `[TYPE = AUTOMATICALLY CONFIGURED, NAME ="name"]`

  where *name* is the name of the peer MTA's entry in the directory.

– `[TYPE = MANUALLY CONFIGURED, NAME ="peer-mta-name"]`

  where *peer-mta-name* is the name of the Peer MTA entity that represents the peer MTA in the other routing domain.

Note that the Interruption Reason attribute only contains information about the association failure when the value of the State attribute is Interrupted.

16.2 Events Relating to Problems with Associations and the Transport Service

The following events relate to problems with associations:

- Outbound Establishment Failure (Section 16.2.1)
- Outbound Hard Rejection (Section 16.2.2)
- Inbound Transfer Hard Rejection (Section 16.2.3)
- Outbound Soft Rejection (Section 16.2.4)
- Inbound Transfer Soft Rejection (Section 16.2.5)
- Inbound Failure (Section 16.2.6)
- Outbound Failure (Section 16.2.7)
- Transport Interface Error (Section 16.2.8)

Note

Events relating to associations can occur when interworking with a peer MTA in another routing domain. To solve the problems indicated by these events, it may be necessary for you to contact the person responsible for managing the peer MTA.

To make it easier to contact a manager in another routing domain, set up the Contact Name attribute of each Peer MTA entity that you create. See Section 14.1 for an example of the command you use to set this attribute.

To solve problems with associations, it may be necessary to stop and start the MTA that generated the event or a peer MTA in your routing domain that is named in the event. The procedures for stopping and starting an MTA are described in the appropriate appendix.

16.2.1 Outbound Establishment Failure

This event occurs whenever an attempt by an MTA to establish an association to a peer MTA fails. The association can fail because of an error detected by either the MTA that initiated the association or by the receiving peer MTA.

It is the responsibility of the MTA that generated this event to make further attempts to establish the association and transfer the MPDU. The MTA makes another attempt to establish the association after the retry interval has elapsed.

This event is counted by the Outbound Establishment Failures counter of the Peer MTA entity that holds information about the peer MTA.

If the attempt to establish the association was aborted by the MTA that generated the event, then the event contains:

```
RTSE Entity Initiating Failure = This MTA
```

This is followed by one of:

1
```
Reason Code = Local System Problem
Local Diagnostic = Internal Error
Association Type = type
```

2
```
Reason Code = Local System Problem
Local Diagnostic = Local Timeout Period Expired
Association Type = type
```

3
```
Reason Code = Temporary Problem
Local Diagnostic = This MTA Disabled
Association Type = type
```

4
```
Reason Code = Temporary Problem
Local Diagnostic = Peer MTA Entity Disabled
Association Type = type
```

5
```
Reason Code = Temporary Problem
Local Diagnostic = Peer MTA Unavailable
Association Type = type
```

6
```
Reason Code = Permanent Problem
Local Diagnostic = Transport Disconnect Received
Association Type = type
```

7
```
Reason Code = Permanent Problem
Local Diagnostic = Invalid MTA Password
Association Type = type
```

8 `Reason Code = Permanent Problem`
`Local Diagnostic = Unrecognized MTA Name`
`Association Type =` *type*

9 `Reason Code = Permanent Problem`
`Local Diagnostic = Mismatch Between Called Address and either`
`Transport Service or OSI Templates`
`Association Type =` *type*

10 `Reason Code = Permanent Problem`
`Local Diagnostic = Invalid Presentation Context Supplied`
`Association Type =` *type*

11 `Reason Code = Protocol Violation`
`Local Diagnostic = Not Applicable`
`Association Type =` *type*

12 `Reason Code = Invalid Parameter`
`Local Diagnostic = No Diagnostic Available`
`Association Type =` *type*

13 `Reason Code = Validation Error`
`Local Diagnostic = Failed to Decode Bind Information`
`Association Type =` *type*

If the attempt to establish the association was aborted by the peer MTA, then the event contains the following:

14 `RTSE Entity Initiating Failure = Peer MTA`
`Reason Code = Permanent Problem`
`Local Diagnostic = No Diagnostic Available`
`Association Type =` *type*

type can be New or Recover according to whether the association is new, or being recovered.

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 The establishment of the association failed because the MTA was affected by an error in its own software. Check the event sink for an occurrence of

the Internal Error event from this MTA and take the appropriate action as described in Section 22.2.1.

2 The establishment of the association failed because the peer MTA took too long to respond. The time limit that specifies how long the MTA can wait for a response from the peer MTA expired. Note that it is not possible to change this time limit. Contact the manager of the peer MTA and find out why the peer MTA did not respond.

3 The establishment of the association failed because the MTA was disabled after the MTA started to set up the association.

Display the current state of the MTA, using the following command:

```
SHOW NODE "node-id" MTA STATE
```

If the MTA is ON, the MTA was re-enabled after the event occurred. In this case, no further action is required. If the MTA is OFF, enable the MTA using the following command:

```
ENABLE NODE "node-id" MTA
```

The MTA makes another attempt to establish the association when you enable it.

4 The establishment of the association failed because the Peer MTA entity that represents the peer MTA was disabled after the MTA started to set up the association.

Display the current state of the Peer MTA entity, using the following command:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] STATE
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

If the Peer MTA entity is ON, the Peer MTA entity was re-enabled after the event occurred. In this case, no further action is required. If the Peer MTA entity is OFF, enable the Peer MTA entity, using the following command:

```
ENABLE NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"]
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

The MTA makes another attempt to establish the association. Retry attempts by the MTA can only succeed if the Peer MTA entity is in the ON state.

5 The establishment of the association failed because the peer MTA is not reachable. The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  Check that OSI Transport is running at the peer MTA's node, using the following command:

  ```
  SHOW NODE "node-id" OSI TRANSPORT STATE
  ```

  where *node-id* is the name of the node where the peer MTA is running.

  If OSI Transport is not running on the peer MTA's node, then restart it, and stop and start the peer MTA.

  If OSI Transport is running, then check that the peer MTA is running, using the following command:

  ```
  SHOW NODE "node-id" MTA STATE
  ```

  where *node-id* is the name of the node where the peer MTA is running.

  If the peer MTA does not exist, then make further investigations at the node where the peer MTA is located.

  If the peer MTA is OFF, then enable it using the following command:

  ```
  ENABLE NODE "node-id" MTA
  ```

  where *node-id* is the name of the node where the peer MTA is running.

  If the peer MTA is running, then there is a discrepancy between the peer MTA's Presentation address held in the directory and the Presentation address being used currently by the peer MTA.

  Note that this discrepancy can only occur if the peer MTA's Presentation address in the directory is changed while the peer MTA is running.

  You need to update the Presentation address used by the peer MTA. To do this, log into a privileged account on the node where the peer MTA is located. Stop the peer MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the peer MTA.

- The peer MTA is in another routing domain.

  Contact the person responsible for managing the peer MTA and find out if the peer MTA is running. If the peer MTA is running, then validate its Presentation address, or Session address if it has one. Check that the peer MTA's address held by the Peer MTA entity that represents the peer MTA is correct. Use the following command to display the peer MTA's address:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] address
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA and *address* is either PRESENTATION ADDRESS or SESSION ADDRESS.

If the peer MTA's Presentation address, or Session address if it has one, is incorrect, then correct the error by setting the Peer MTA entity's Presentation Address or Session Address attribute.

See the DECnet/OSI network documentation for more information about Presentation and Session addresses.

6 The establishment of the association failed because of an error in one of the lower layers, for example the Transport layer, or because the physical line broke.

If this error occurs frequently because of an error in one of the lower layers, then investigate it further using the lower layer management entities. See the HP DECnet-Plus documentation for information about the lower layer management entities.

7 The establishment of the association failed because the MTA did not recognize the password supplied by the peer MTA. The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  In this case there is a discrepancy between the password supplied by the peer MTA and the peer MTA's password held in the directory.

  Note that this discrepancy can only occur if the peer MTA's password in the directory is changed while the peer MTA is running.

  You need to update the password used by the peer MTA. To do this, log into a privileged account on the node where the peer MTA is located. Stop the peer MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the peer MTA.

- The peer MTA is in another routing domain.

  In this case, there is a discrepancy between the password supplied by the peer MTA and the password held in the Peer MTA entity that represents the peer MTA. Contact the person managing the peer MTA in the other routing domain and verify the peer MTA's password.

The peer MTA's password is held in the Peer Password attribute of the Peer MTA entity. This is a write-only characteristic attribute, so you cannot display its value. If you have included the commands that create and set up the Peer MTA entity in the MTA's startup script, then you can obtain the password from this script. Refer to the appendix describing the operating system specific information for the location of the MTA's startup script.

Use the following command to reset the password:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER PASSWORD password
```

where:

- *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.
- *password* is one of the following:
  - `[TYPE = IA5, IA5 String = "string"]`
  - `[TYPE = Octet, Octet String = 'string'h]`

    where *string* is the peer MTA's password in either IA5 graphic subset characters or hexadecimal octets.

8 The establishment of the association failed because the MTA did not recognize the peer MTA's name. The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  In this case, there is a discrepancy between the name supplied by the peer MTA and the peer MTA's name held in the directory.

  Note that this discrepancy can only occur if the name of the peer MTA's entry in the directory is changed while the peer MTA is running.

  You need to update the name used by the peer MTA. To do this, log into a privileged account on the node where the peer MTA is located. Stop the peer MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the peer MTA.

- The peer MTA is in another routing domain.

  In this case there is a discrepancy between the name supplied by the peer MTA and the name held by the Peer MTA entity that represents the peer MTA. Contact the person managing the peer MTA in the other routing domain and verify the peer MTA's name.

The peer MTA's name is held in the Peer Name attribute of the Peer MTA entity. Display the Peer Name attribute and check that there is an error. Use the following command to reset its value:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER NAME "name"
```

where `peer-mta-name` is the name of the Peer MTA entity that holds information about the peer MTA, and `name` is the name of the peer MTA.

9 The establishment of the association failed because the peer MTA's address does not contain an NSAP that corresponds to the type of Transport Service used by the MTA for connections to the peer MTA.

For outbound connections to a peer MTA in the same routing domain, the MTA uses the Transport Service specified by the Transport Service Options attribute of the MTA entity. The Transport Service Options attribute can contain one or both of the following values:

- OSI

  If the Transport Service Options attribute contains this value, the MTA can use the DECnet/OSI Transport Service.

- TCPIP

  If the Transport Service Options attribute contains this value, the MTA can use the TCP/IP Transport Service.

Similarly, the Peer MTA entity contains a Transport Service Options attribute. You can use this attribute to specify the Transport Service used by the boundary MTA for outbound connections to a peer MTA in another routing domain. If the Transport Service Options attribute of a Peer MTA entity is set to null, the boundary MTA uses the Transport Service specified by the Transport Service Options attribute of the MTA entity.

In order for an MTA to make a connection to the peer MTA, the peer MTA's Presentation or Session address must contain an NSAP that corresponds to the Transport Service used by the MTA.

If an MTA uses the DECnet/OSI Transport Service for outbound connections to a peer MTA, the following also apply:

- The peer MTA's Presentation or Session address must contain either a CLNS or CONS NSAP.
- The MTA must use a Transport Template entity that specifies a network service that matches the CLNS or CONS NSAP is the peer MTA's address.

The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  Check the value or values specified by the MTA entity's Transport Service Options attribute. This attribute is set up in the MTA's startup script; however, you may have altered its value since you set up the MTA. Use the following command to display the value of this attribute:

  ```
  SHOW NODE "node-id" MTA TRANSPORT SERVICE OPTIONS
  ```

  Check that the peer MTA's Presentation and Session addresses contain an NSAP that corresponds to the Transport Service specified by the MTA. To do this, issue the following command at the node where the peer MTA is running:

  ```
  SHOW NODE node-id MTA PRESENTATION ADDRESS, SESSION ADDRESS
  ```

  Correct any discrepancy between the Transport Service used by the MTA and the NSAPs in the peer MTA's address.

  If the MTA's Transport Service Options attribute specifies OSI, and the peer MTA's address contains a CLNS or CONS NSAP, the Transport Service is unable to find a corresponding Transport Template entity.

  Display the MTA's Template Name attribute to find out what Template entities it specifies:

  ```
  SHOW NODE "node-id" MTA TEMPLATE NAME
  ```

  The Template Name attribute is set up in the MTA's startup script and the corresponding Template entities are created by the MTA installation and setup procedures. You might have modified the Template Name attribute to name a different Transport Template entity or reset the attribute to be null. See Section 7.7.5 for information about the implications of setting the Template Name attribute to null.

  Check that the Transport Template entity specified by the Template Name attribute exists on the node where the MTA is running and that the value of its Network Service attribute corresponds to an NSAP in the peer MTA's address. See Section 7.7.4 for information about the MTA's Transport Template entities. See the DECnet/OSI network documentation for information about creating and modifying Transport Template entities.

- The peer MTA is in another routing domain.

  In this case, check the value of the Peer MTA entity's Transport Service Options attribute. Use the following command to display the value of this attribute:

  ```
  SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY -
  CONFIGURED, NAME = "peer-mta-name"] TRANSPORT SERVICE OPTIONS
  ```

  where *peer-mta-name* is the name of the Peer MTA entity.

  If the Peer MTA entity's Transport Service Options attribute does not contain a value, the boundary MTA is using the Transport Service specified its MTA entity. In this case, check the value of the MTA entity's Transport Service Options attribute, as described in the action for a peer MTA in your routing domain.

  Find out if the peer MTA's Presentation or Session address contains an NSAP that corresponds to the Transport Service being used by the boundary MTA for outbound connections to the peer MTA. Use the following command to display the peer MTA's address:

  ```
  SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
  NAME = "peer-mta-name"] PRESENTATION ADDRESS, SESSION ADDRESS
  ```

  where *peer-mta-name* is the name of the Peer MTA entity.

  Note that the Peer MTA entity has either a Presentation Address or Session Address attribute, not both; this command displays whichever attribute is present.

  If the peer MTA's address does not contain an NSAP that corresponds to the Transport Service used by the boundary MTA, contact the person responsible for managing the peer MTA and verify the peer MTA's address. Make sure that the boundary MTA is using the correct address for the peer MTA and that the Transport Service used by the boundary MTA corresponds to an NSAP in the peer MTA's address. Refer to the DECnet/OSI network documentation for more information about Presentation and Session addresses.

  If the Transport Service Options attribute of the Peer MTA entity or the MTA entity, whichever is being used, specifies OSI and the peer MTA's Presentation or Session address contains a CLNS or CONS NSAP, the Transport Service is unable to find a corresponding Template entity.

  Use the following command to display what Transport Template entities are specified by the Peer MTA entity:

  ```
  SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
  NAME = "peer-mta-name"] TEMPLATE NAME
  ```

  where *peer-mta-name* is the name of the Peer MTA entity.

If the Template Name does not specify a Transport Template entity, then the MTA is using the Transport Template entities specified by the Template Name attribute of the MTA entity. Check the value of the MTA entity's Template Name attribute as described in the action for a peer MTA in your routing domain.

If you have tuned the Peer MTA entity so that it is specifying a Transport Template entity that you have created, make sure that the value of the Transport Template entity's Network Service attribute corresponds to an NSAP in the peer MTA's address. See the DECnet/OSI network documentation for information about modifying OSI Transport Template entities.

10 The establishment of the association failed because of a protocol error in the interface between the RTSE and the Presentation layer. Specifically, one or more Presentation Context Identifiers sent in the RT-OPEN response protocol from the peer MTA is invalid.

This is due to an error in the peer MTA's software. Inform the manager of the peer MTA about the problem. To trace the error, record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

11 The establishment of the association failed because the MTA detected a protocol error in one of its lower layers.

Check the event sink for an occurrence of the Lower Layer Protocol Violation event (see Section 16.3.2) or the RTSE Protocol Violation event (see Section 16.3.1) from this MTA. These events identify the source of the protocol error. If the information provided by these events is insufficient to identify the protocol error, then record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

12 The establishment of the association failed because the MTA received a parameter from the peer MTA that it did not recognize. Inform the manager of the peer MTA about the problem.

To identify the parameter, record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

13 The MTA is unable to decode the ASN.1 data part of the name or password supplied by the peer MTA.

Record the protocol information that is exchanged between the two MTAs as described in Section 16.4 and inform the person managing the peer MTA about the problem.

If, after you have consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

14 The failure is due to a problem reported by the peer MTA.

Contact the manager of the peer MTA who should be able to explain why the peer MTA failed to accept the association.

16.2.2 Outbound Hard Rejection

This event occurs whenever an attempt by an MTA to establish an association to a peer MTA is rejected by the peer MTA because of a permanent problem. For example, the peer MTA rejects the association because it does not recognize the MTA's password. Occurrences of this event must be investigated and solved before an attempt to establish an association to the peer MTA can succeed.

This event is counted by the Outbound Hard Rejections counter of the Peer MTA entity that holds information about the peer MTA.

The information provided by the event is one of the following:

1 `Reason Code = Unacceptable Dialogue Mode`

2 `Reason Code = Validation Error`

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 The peer MTA is rejecting RTSE monologue dialogue mode. Contact the manager of the peer MTA and ensure that the peer MTA is able to accept RTSE monologue dialogue mode.

2 The peer MTA does not recognize the MTA's name or password. The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  In this case, there is a discrepancy between the name or password supplied by the MTA and the MTA's name or password held in the directory.

  Note that this discrepancy can only occur if the name of the MTA's entry in the directory or the password in the MTA's entry is changed while the MTA is running.

  You need to update the name or password used by the MTA. To do this, log into a privileged account on the node where the MTA is located. Stop the MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the MTA.

- The peer MTA is in another routing domain.

  In this case, there is a discrepancy between the name or password supplied by the MTA and the MTA's name or password expected by the peer MTA.

  Contact the person managing the peer MTA and verify that the peer MTA is using the correct name and password of the MTA.

  The name and password supplied by the MTA is held in the Peer MTA entity that holds information about the peer MTA. The MTA's name is held in the Local Name attribute and the MTA's password is held in the Local Password attribute.

  Display the Local Name attribute and check that there is no obvious error. Use the following command to display this attribute:

  ```
  SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
  NAME = "peer-mta-name"] LOCAL NAME
  ```

  where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

If there is an error, then use the following command to reset this attribute:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] LOCAL NAME "name"
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA and *name* is the name of the boundary MTA.

The Local Password attribute is a write-only characteristic attribute, so you cannot display its value. If you have included the commands that create and set up the Peer MTA entity in the MTA's startup script, then you can obtain the password from this script. Refer to the appendix describing the operating system specific information for the location of the MTA's startup script.

Use the following command to reset the password:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] LOCAL PASSWORD password
```

where:

- *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.
- *password* is one of the following:
  - `[TYPE = IA5, IA5 String = "string"]`
  - `[TYPE = Octet, Octet String = 'string'h]`

    where *string* is the peer MTA's password in either IA5 graphic subset characters or hexadecimal octets.

16.2.3 Inbound Transfer Hard Rejection

This event occurs whenever an MTA rejects an attempt by a peer MTA to establish an association because of a permanent problem. For example, the MTA rejects the association because it does not recognize the peer MTA's password. Occurrences of this event must be investigated and solved before an attempt by the peer MTA to establish an association with the MTA can succeed.

This event is counted by the MTA entity's Inbound Transfer Hard Rejections counter.

The event provides the name and address of the peer MTA that initiated the association:

```
Calling MTA Name = mta-name
Calling Address = address
```

where *mta-name* is the name of the peer MTA that initiated the association and *address* is the address of that peer MTA.

This is followed by one of:

```
1  Reason Code = Unacceptable Dialogue Mode
   Local Diagnostic = Not Applicable

2  Reason Code = Invalid Application Context
   Local Diagnostic = Not Applicable

3  Reason Code = Validation Error
   Local Diagnostic = Unrecognized MTA Name

4  Reason Code = Validation Error
   Local Diagnostic = Unknown Peer MTA

5  Reason Code = Validation Error
   Local Diagnostic = Invalid MTA Password

6  Reason Code = Validation Error
   Local Diagnostic = Invalid Presentation Context Supplied

7  Reason Code = Validation Error
   Local Diagnostic = Invalid Direction of Transfer

8  Reason Code = Validation Error
   Local Diagnostic = Unrecognized MTA Calling Address

9  Reason Code = Validation Error
   Local Diagnostic = Failed to Decode Bind Information

10 Reason Code = Validation Error
   Local Diagnostic = Peer Domain not Configured in the Directory
```

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 The peer MTA is requesting RTSE two-way-alternate dialogue mode. This dialogue mode is not supported by the MTA. Contact the manager of the peer MTA and ensure that the peer MTA is able to request RTSE monologue dialogue mode.

2 There is a problem with the peer MTA's software. The peer MTA has sent an incorrect object identifier for the application context in the RT-OPEN request that does not conform to the MTS Transfer protocol. Notify the manager of the peer MTA about the problem.

If, after you have consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide.

3 The peer MTA's name is not recognized by the MTA. The peer MTA's name is provided in the Calling MTA Name field in the event. There is a discrepancy between the name supplied in the event and the name held in the Peer Name attribute of the Peer MTA entity that represents the peer MTA.

Display the Peer Name attribute and check if there is an obvious error. Use the following command to display this attribute:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER NAME
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

If there is no obvious error in the value of this attribute, contact the person managing the peer MTA in the other routing domain and verify the peer MTA's name.

If the name used by the peer MTA is incorrect, then it is the responsibility of the person managing the peer MTA to correct it. If the name of the peer MTA held by the Peer MTA entity is incorrect, then reset the Peer Name attribute. Use the following command to set this attribute:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER NAME "name"
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA and *name* is the amended name of the peer MTA that it uses in its communication.

4 The MTA is unable to validate the peer MTA's name and password. The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  In this case, the MTA is unable to find information about the peer MTA in the directory.

  Note that this problem can only occur if the peer MTA's entry was deleted or modified in the directory while the peer MTA is running.

Recreate the peer MTA's entry in the directory. To do this, log into a privileged account on the node where the peer MTA is located. Stop the peer MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the peer MTA.

- The peer MTA is in another routing domain.

  Try to find a Peer MTA entity with the Peer Name attribute that matches the name supplied in the Calling MTA Name field in the event. To do this, use the following command:

  ```
  SHOW NODE "node-id" MTA PEER MTA *, WITH PEER -
  NAME = "calling-mta-name"
  ```

  where *calling-mta-name* is the name provided in the Calling MTA Name field in the event.

  If this command displays the identifier of a Peer MTA entity, then check the Presentation and Session Address attributes of that entity. Use the following command to display these attributes:

  ```
  SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
  NAME = "peer-mta-name"] PRESENTATION ADDRESS, SESSION ADDRESS
  ```

  where *peer-mta-name* is the name of the Peer MTA entity.

  Note that the Peer MTA entity has either a Presentation Address or Session Address attribute, not both; this command displays whichever attribute is present.

  Check with the manager of the peer MTA and find out why the peer MTA is using a different address from the one that you have entered in the Peer MTA entity. Ensure that the set of network service access points (NSAPs) which the peer MTA can use are included in the peer MTA's address.

  If the peer MTA's Presentation address, or Session address if it has one, is incorrect, then correct the error by setting the Peer MTA entity's Presentation Address or Session Address attribute.

  See the DECnet/OSI network documentation for more information about Presentation and Session addresses.

  If you are unable to find a Peer MTA entity, use the peer MTA's name and address provided by the event to identify the peer MTA that is trying to connect to the MTA. Find out if the peer MTA should be trying to connect to the MTA. If the peer MTA should not be trying to connect to the MTA, investigate why the peer MTA is making the attempt.

If you want the peer MTA to connect to the MTA, then create and set up a Peer MTA entity to hold information about the peer MTA. See *HP MAILbus 400 MTA Planning and Setup* for information about planning a Peer MTA entity.

Another possible cause of this event is that the Peer MTA entity you created to represent the peer MTA has been deleted by mistake. When you delete a Peer MTA entity the MTA generates the Entity Deleted event. To find out if the Peer MTA entity has been deleted, check the event sink for an Entity Deleted event.

If the Peer MTA entity had previously existed, check the MTA's startup script to see if you have entered the create command and the commands that set up its characteristic attributes. Refer to the appendix describing the operating system specific information for the location of this script. If the script contains the appropriate commands, then use their values to set up the Peer MTA entity.

5 The peer MTA's password is not recognized by the MTA. The action that you take depends on whether the peer MTA is in your routing domain or in another routing domain:

- The peer MTA is in your routing domain.

  In this case, there is a discrepancy between the password supplied by the peer MTA and the peer MTA's password held in the directory.

  Note that this discrepancy can only occur if the peer MTA's password in the directory is changed while the peer MTA is running.

  You need to update the password used by the peer MTA. To do this, log into a privileged account on the node where the peer MTA is located. Stop the peer MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the peer MTA.

- The peer MTA is in another routing domain.

  In this case, there is a discrepancy between the password supplied by the peer MTA and the password held in the Peer MTA entity that represents the peer MTA. Contact the person managing the peer MTA in the other routing domain and verify the peer MTA's password.

  The peer MTA's password is held in the Peer Password attribute of the Peer MTA entity. This is a write-only characteristic attribute, so you cannot display its value. If you have included the commands that create and set up the Peer MTA entity in the MTA's startup script, then you can obtain the password from this script. Refer to the appendix describing the operating system specific information for the location of the MTA's startup script.

Use the following command to reset the password:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER PASSWORD password
```

where:

- *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.
- *password* is one of the following:
    - `[TYPE = IA5, IA5 String = "string"]`
    - `[TYPE = Octet, Octet String = 'string'h]`

      where *string* is the peer MTA's password in either IA5 graphic subset characters or hexadecimal octets.

6 The MTA rejected the association because of a protocol error in the interface between the RTSE and the Presentation layer. Specifically, one or more Presentation Context Identifiers sent in the RT-OPEN response protocol from the peer MTA is invalid.

This is due to an error in the peer MTA's software. Inform the manager of the peer MTA about the problem.

If, after you have consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide.

7 The MTA rejected the association because inbound associations from the peer MTA are not permitted.

The permitted direction of MPDU transfer between the MTA and the peer MTA is specified in the Direction attribute of the Peer MTA entity that represents the peer MTA. You can set this attribute to one of the following values:

- Outbound
- Inbound
- Inbound and Outbound

In this case, the Direction attribute is set to outbound.

Check with the manager of the peer MTA and find out why the peer MTA is trying to send a message to the MTA. If the peer MTA is incorrectly configured, then the peer MTA manager is responsible for correcting the error. If you want the peer MTA to send messages to the MTA, then reset the Peer MTA entity's Direction attribute. Use the following command to set this attribute:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] DIRECTION direction
```

where *peer-mta-name* is the name of the Peer MTA entity and *direction* is the direction of MPDU transfer that you want the boundary MTA to use for this particular peer MTA.

8 The MTA does not recognize the peer MTA's address. The Calling Address field of the event contains the address of the peer MTA.

There is a discrepancy between the peer MTA's Presentation address held in the directory and the Presentation address being used currently by the peer MTA.

Note that this discrepancy can only occur if the peer MTA's Presentation address in the directory is changed while the peer MTA is running.

You need to update the Presentation address used by the peer MTA. To do this, log into a privileged account on the node where the peer MTA is located. Stop the peer MTA, then follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the peer MTA.

9 The MTA is unable to decode the ASN.1 data part of the name or password supplied by the peer MTA. Record the protocol information that is exchanged between the two MTAs as described in Section 16.4 and inform the person managing the peer MTA about the problem.

If, after you have consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

10 The MTA is unable to find a Domain entry in the directory that represents the routing domain where the peer MTA is located. The MTA was looking in the directory for a Domain entry that matched the value of the Peer Domain attribute of the Peer MTA entity that represents the peer MTA.

The error can be caused by one of the following:

- The name of the Domain entry specified in the Peer Domain attribute is incorrect.

- The Domain entry in the directory is incorrectly named or missing.

When this error occurs, the MTA also generates the Unknown Peer Domain event.

Find out the correct name of the Domain entry in the directory that represents the domain where the peer MTA is located. The correct name is the name that was planned. Check that there is a Domain entry in the directory with the correct name. Use the following command:

```
SHOW  "/MTS=routing-domain-name" DOMAIN "domain-name" -
ALL ATTRIBUTES
```

where:

- *routing-domain-name* is the name of your routing domain
- *domain-name* is the correct name of the Domain entity that represents the routing domain where the peer MTA is located

Display the Peer Domain attribute of the Peer MTA entity that holds information about the peer MTA. Use the following command to display this attribute:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER DOMAIN
```

where *peer-mta-name* is the name of the Peer MTA entity that represents the peer MTA.

Ensure that there is no discrepancy between the value of the Peer Domain attribute and the name of the Domain entity. If necessary, correct the Peer Domain attribute in the Peer MTA entity using the following command:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] PEER DOMAIN "domain-name"
```

where *peer-mta-name* is the name of the Peer MTA entity that represents the peer MTA.

If the Domain entry does not exist in the directory, then create one. See the *MTS Module Online Help* for information about creating and setting up a Domain entity.

Make sure that the Peer Domain attribute in the Peer MTA entity contains the name of the Domain entity that you create in the directory.

16.2.4 Outbound Soft Rejection

This event occurs whenever an attempt by an MTA to establish an association is rejected by a peer MTA because of a temporary problem. For example, the peer MTA rejects the association because it is congested.

The MTA that initiated the association makes another attempt to establish the association after the retry interval has expired.

This event is counted by the Outbound Soft Rejections counter of the Peer MTA entity that holds information about the peer MTA.

The event contains one of the following reasons for the rejection:

1 `RTSE Reason Code = Cannot Recover`

2 `RTSE Reason Code = RTSE Busy`

3 `RTSE Reason Code = Lower Layer Reject Received`

4 `RTSE Reason Code = No Reason Supplied`

5 `RTSE Reason Code = Invalid Reason Supplied`

Action

1 A previous association from the MTA failed while an MPDU was being transferred to the peer MTA. The peer MTA is unable to accept the reconnection attempt because it does not have information that allows the transfer to be resumed.

Note that in this case the Lower Layer Reason Code is not significant; no action is required. The MTA initiates a new association and re-sends the complete MPDU.

2 The peer MTA is congested.

The MTA continues to try to establish an association with the peer MTA after the retry interval has elapsed. The attempt can succeed only when the peer MTA is able to accept new inbound associations; therefore, no action is required. However, if a particular peer MTA in your routing domain frequently rejects associations from the MTA, then tune the peer MTA so that it can accept more associations concurrently (see Section 7.3.1).

If a peer MTA in another routing domain frequently rejects associations from the MTA, then notify the person responsible for managing the peer MTA. If possible, the peer MTA needs to be tuned to accept more associations concurrently. Alternatively, increase the number of MTAs in the other routing domain that the boundary MTA can communicate with. Note that in this case the Lower Layer Reason Code is not significant.

3 One of the lower layers of the peer OSI implementation has rejected your MTA's request for an association. In this case the Lower Layer Reason Code gives further details. Pass this information on to the manager of the appropriate lower layer.

4 If one of the lower layers of the peer OSI implementation rejects the MTA's request for an association without supplying a reason, pass this information on to the manager of the appropriate lower layer.

5 If one of the lower layers of the peer OSI implementation rejects the MTA's request for an association, giving an invalid reason, pass this information on to the manager of the appropriate lower layer.

16.2.5 Inbound Transfer Soft Rejection

This event occurs whenever an MTA rejects an attempt by a peer MTA to establish an association because of a temporary problem. For example, the MTA rejects the association because it is congested.

The peer MTA is responsible for setting up the association. If the peer MTA is a MAILbus 400 MTA, then it makes another attempt to establish the association after the retry interval has elapsed. If the peer MTA is not a MAILbus 400 MTA, then contact the manager of the peer MTA and find out how it is likely to respond.

This event is counted by the MTA entity's Inbound Transfer Soft Rejections counter.

The event provides the name and address of the peer MTA that initiated the association:

```
Calling MTA Name = mta-name
Calling Address = address
```

where *mta-name* is the name of the peer MTA and *address* is the address of the peer MTA.

This is followed by one of:

1
```
Reason Code = Cannot Recover
Local Diagnostic = No Diagnostic Available
```

2
```
Reason Code = RTSE Busy
Local Diagnostic = Maximum Associations Reached
```

3
```
Reason Code = RTSE Busy
Local Diagnostic = Maximum Inbound Transfer Associations Reached
```

4 `Reason Code = RTSE Busy`
`Local Diagnostic = Maximum Inbound Parallel Transfer Associations Reached`

5 `Reason Code = RTSE Busy`
`Local Diagnostic = MTA Disabling`

6 `Reason Code = RTSE Busy`
`Local Diagnostic = Peer MTA Entity Disabled`

7 `Reason Code = RTSE Busy`
`Local Diagnostic = Directory Service Error`

8 `Reason Code = RTSE Busy`
`Local Diagnostic = Local System Problem`

9 `Reason Code = RTSE Busy`
`Local Diagnostic = Insufficient Resources`

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 A previous association from the peer MTA failed while an MPDU was being transferred to the MTA. The MTA is unable to accept the reconnection attempt because it does not have information that allows the transfer to be resumed.

Normally, the MTA resumes message transfer when a peer MTA requests an association recovery. However, this error means that the MTA has been deleted and then recreated since the initial transfer attempt. The peer MTA initiates a new association and re-sends the complete MPDU; therefore, no action is required.

2 The maximum number of associations to and from peer MTAs that the MTA can have at any one time has been reached. The maximum number of concurrent associations is specified by the Maximum Transfer Associations attribute of the MTA entity.

Retry attempts by the peer MTA can only succeed when the MTA is able to accept new associations. If this problem persists, tune the MTA so that it can accept more inbound associations concurrently (see Section 7.3.1).

If this problem persists and the peer MTA is in another routing domain, increase the number of boundary MTAs that communicate with the other routing domain. This allows more messages from the other routing domain to enter your routing domain concurrently.

3 The maximum number of inbound associations from any number of peer MTAs that the MTA can have at any one time has been reached. The maximum number of concurrent associations is specified by the Maximum Inbound Transfer Associations attribute of the MTA entity.

Retry attempts by the peer MTA can only succeed when the MTA is able to accept new associations. If this problem persists, tune the MTA so that it can accept more inbound associations concurrently (see Section 7.3.1).

If this problem persists and the peer MTA is in another routing domain, increase the number of boundary MTAs that communicate with the other routing domain. This allows more messages from the other routing domain to enter your routing domain concurrently.

4 The maximum number of associations that the MTA can have at any one time from a specific peer MTA located in another routing domain has been reached. The maximum number of concurrent associations from a specific peer MTA in another routing domain is specified by the Maximum Inbound Parallel Transfer Associations attribute of the associated Peer MTA entity.

Retry attempts by the peer MTA can only succeed when the MTA is able to accept new associations. If this problem persists, tune the MTA so that it can accept more inbound associations concurrently from the specific peer MTA (see Section 7.3.1).

5 The MTA has recently been disabled. When the peer MTA tried to set up the association, the Disable command had not completed and the MTA was in the DISABLING state.

Display the current state of the MTA, using the following command:

```
SHOW NODE "node-id" MTA STATE
```

If the MTA is ON, the MTA was re-enabled after the event occurred. In this case, no further action is required. If the MTA is OFF, enable the MTA using the following command:

```
ENABLE NODE "node-id" MTA
```

If the MTA is in any other state, or does not respond to NCL commands, then stop the MTA and restart it as explained in the appendix describing the operating system specific information.

Retry attempts by the peer MTA can only succeed when the MTA is in the ON state.

6 When the peer MTA tried to set up the association, the Peer MTA entity that holds information about the peer MTA was disabled. Consequently, the MTA was unable to access the information that it needed to communicate with the peer MTA.

Display the current state of the Peer MTA entity, using the following command:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] STATE
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

If the Peer MTA entity is ON, the Peer MTA entity was re-enabled after the event occurred. In this case, no further action is required. If the Peer MTA entity is OFF, enable the Peer MTA entity using the following command:

```
ENABLE NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"]
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

Retry attempts by the peer MTA can only succeed if the Peer MTA entity is in the ON state.

7 The MTA rejected the association because it was unable to access the directory and is therefore unable to validate the peer MTA's name or password.

Check the event sink for occurrences of the Directory Service Error event from this MTA and take the appropriate action as described in Section 20.2.

Retry attempts by the peer MTA can only succeed when the MTA is able to access the directory.

8 The MTA rejected the association because of a temporary problem at the MTA. Check the event sink for an occurrence of the System Interface Error event from this MTA and take the appropriate action as described in Section 20.1.

If no such event exists, then disable and re-enable the MTA. If you cannot resolve the problem, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to supply.

9 The MTA rejected the association because of a problem with one or more of its resources. This is a temporary problem and the MTA should accept the association when resources become available. If the MTA continues to reject the association, then further events should indicate the problem at the MTA.

16.2.6 Inbound Failure

This event occurs whenever an association initiated by a peer MTA is aborted. The association can be aborted by either the MTA or the peer MTA.

This event is counted by the Inbound Failures counter of the Peer MTA entity that holds information about the peer MTA.

The peer MTA is responsible for recovering the association. If the peer MTA is a MAILbus 400 MTA, then it always attempts to recover the association. If the peer MTA is not a MAILbus 400 MTA, then contact the manager of the peer MTA and find out how it is likely to respond.

The event identifies the MTA that aborted the association and provides information about the cause of the event.

If the association was aborted by the MTA that generated the event, then the event contains:

```
RTSE Entity Initiating Failure = This MTA
```

This is followed by one of:

1
```
Abort Reason Code = Local System Problem
Local Diagnostic = Internal Error
```

2
```
Abort Reason Code = Local System Problem
Local Diagnostic = Local Timeout Period Expired
```

3
```
Abort Reason Code = Local System Problem
Local Diagnostic = Inbound Idle Timer Expired
```

4
```
Abort Reason Code = Local System Problem
Local Diagnostic = Insufficient Resources
```

5
```
Abort Reason Code = Local System Problem
Local Diagnostic = System Interface Error
```

6
```
Abort Reason Code = Temporary Problem
Local Diagnostic = This MTA Disabled
```

7
```
Abort Reason Code = Temporary Problem
Local Diagnostic = Peer MTA Entity Disabled
```

8
```
Abort Reason Code = Temporary Problem
Local Diagnostic = Activity Deleted
```

9
```
Abort Reason Code = Permanent Problem
Local Diagnostic = Transport Disconnect Received
```

10 `Abort Reason Code = Protocol Violation`
`Local Diagnostic = Not Applicable`

11 `Abort Reason Code = Invalid Parameter`
`Local Diagnostic = No Diagnostic Available`

12 `Abort Reason Code = Unrecognized Activity`
`Local Diagnostic = No Diagnostic Available`

13 `Abort Reason Code = Transfer Completed`
`Local Diagnostic = No Diagnostic Available`

If the association was aborted by the peer MTA the event contains:

14
```
RTSE Entity Initiating Failure = Peer MTA
Abort Reason Code = reason
Local Diagnostic = No Diagnostic Available
```

where *reason* is one of the following:

```
Local System Problem
Temporary Problem
Permanent Problem
Protocol Violation
Invalid Parameter
```

All occurrences of this event also contain the number of octets and MPDUs that were transferred from the peer MTA to the MTA before the association failed:

```
Octets In = integer
MPDUs In = integer
```

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 The MTA aborted the association because it was affected by an error in its own software. Check the event sink for an occurrence of the Internal Error event from this MTA and take the appropriate action as described in Section 22.2.1.

2 The MTA aborted the association because the peer MTA took too long to respond. The time limit that specifies how long the MTA can wait for a response from the peer MTA expired. Note that it is not possible to change this time limit. Contact the manager of the peer MTA and find out why the peer MTA did not respond.

3 The MTA aborted the association because the time limit that specifies how long the association can remain idle after an MPDU has been transferred expired. On inbound associations, the association idle time is specified by the MTA entity's Maximum Idle Inbound Transfer Association Interval attribute.

If this problem occurs frequently in your routing domain, then it could be because the MTA's Maximum Idle Inbound Transfer Association Interval attribute is set up incorrectly. See Section 7.3.3.2 for information about how to set this attribute so that associations are released and not aborted.

4 The MTA aborted the association because the operating system could not provide the MTA with sufficient memory resources. When this error occurs, the MTA also generates the System Interface Error event with the error `Memory Allocation`. The System Interface Error event provides more information about the problem. Check the event sink for an occurrence of that event from this MTA and take the appropriate action as described in Section 20.1.

5 The MTA aborted the association because the operating system was unable to provide a service requested by the MTA. When this error occurs, the MTA also generates the System Interface Error event. The System Interface Error event provides more information about the problem. Check the event sink for an occurrence of that event from this MTA and take the appropriate action as described in Section 20.1.

6 The MTA aborted the association because it was disabled after it accepted the association from the peer MTA.

Display the current state of the MTA, using the following command:

```
SHOW NODE "node-id" MTA STATE
```

If the MTA is ON, the MTA was re-enabled after the event occurred. In this case, no further action is required. If the MTA is OFF, enable the MTA using the following command:

```
ENABLE NODE "node-id" MTA
```

An attempt by the peer MTA to recover the association can only succeed when the MTA is in the ON state.

7 The Peer MTA entity that holds information about the peer MTA was disabled after the MTA accepted the association.

Display the current state of the Peer MTA entity, using the following command:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] STATE
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

If the Peer MTA entity is ON, the Peer MTA entity was re-enabled after the event occurred. In this case, no further action is required. If the Peer MTA entity is OFF, enable the Peer MTA entity using the following command:

```
ENABLE NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"]
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

Attempts by the peer MTA to recover the association can only succeed when the Peer MTA entity is in the ON state.

8 The MTA aborted the association because the Activity entity that held information about the association was manually deleted.

An attempt by the peer MTA to recover the association can fail because the recovery information held in the Activity entity has been deleted. A new association from the peer MTA will succeed. No action is required.

9 The association was aborted by one of the lower layers, for example the Transport layer, or because the physical line was broken.

If this error occurs frequently, then investigate it further using the lower layer management entities. See the HP DECnet-Plus documentation for information about the lower layer management entities.

10 The MTA aborted the association because a protocol error was detected in one of its lower layers.

Check the event sink for an occurrence of the Lower Layer Protocol Violation event (see Section 16.3.2) or the RTSE Protocol Violation event (see Section 16.3.1) from this MTA. These events identify the source of the protocol error. If the information provided by these events is insufficient to identify the protocol error, then record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

11 The MTA aborted the association because it received a parameter from the peer MTA that it did not recognize. Inform the manager of the peer MTA about the problem.

To identify the parameter, record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

12 The MTA aborted the association because it did not recognize the recovery protocol information sent by the peer MTA. This can occur when the peer MTA tries to recover a failed association and resume the transfer of an MPDU. If the Activity Identifier in the recovery information does not match what the MTA is expecting, it aborts the association.

There may be a problem at the peer MTA. An isolated occurrence of this event can be ignored because the peer MTA eventually re-sends the complete MPDU. However, if it repeatedly occurs, the manager of the peer MTA needs to verify the recovery behavior of the peer MTA.

13 The MTA aborted the association because it had already received the MPDU that the peer MTA is sending.

No action is required.

14 The abort is due to a problem detected by the peer MTA. Notify the manager of the peer MTA about the problem, and specify the Abort Reason Code. If the peer MTA is a MAILbus 400 MTA, then there should be a corresponding Outbound Failure event generated by the peer MTA. This event provides more information about why the peer MTA aborted the association.

If the failure is due to a protocol violation, it may be necessary to record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

16.2.7 Outbound Failure

This event occurs whenever an association initiated by an MTA to a peer MTA is aborted. The association can be aborted by either the MTA that initiated the association or the peer MTA.

The MTA that initiated the association is responsible for recovering the association. The MTA makes an attempt to recover the association after the retry interval has elapsed.

This event is counted by the Outbound Failures counter of the Peer MTA entity that holds information about the peer MTA.

The event identifies the MTA that aborted the association and provides information about the cause of the event.

If the association was aborted by the MTA that generated the event, then the event contains:

```
RTSE Entity Initiating Failure = This MTA
```

This is followed by one of:

1
```
Abort Reason Code = Local System Problem
Local Diagnostic = Workspace Error
```
2
```
Abort Reason Code = Local System Problem
Local Diagnostic = Internal Error
```
3
```
Abort Reason Code = Local System Problem
Local Diagnostic = System Interface Error
```
4
```
Abort Reason Code = Local System Problem
Local Diagnostic = Local Timeout Period Expired
```
5
```
Abort Reason Code = Temporary Problem
Local Diagnostic = This MTA Disabled
```
6
```
Abort Reason Code = Temporary Problem
Local Diagnostic = Peer MTA Entity Disabled
```
7
```
Reason Code = Temporary Problem
Local Diagnostic = MPDU Deleted
```
8
```
Reason Code = Temporary Problem
Local Diagnostic = Activity Deleted
```
9
```
Abort Reason Code = Permanent Problem
Local Diagnostic = Transport Disconnect Received
```

10 `Abort Reason Code = Protocol Violation`
`Local Diagnostic = Not Applicable`

11 `Abort Reason Code = Invalid Parameter`
`Local Diagnostic = No Diagnostic Available`

If the association was aborted by the peer MTA the event contains:

12 `RTSE Entity Initiating Failure = Peer MTA`
`Abort Reason Code =` *reason*
`Local Diagnostic = No Diagnostic Available`

where *reason* is one of the following:

```
Local System Problem
Temporary Problem
Permanent Problem
Protocol Violation
Invalid Parameter
Unrecognized Activity
Transfer Completed
```

All occurrences of this event also contain the number of octets and MPDUs that were transferred to the peer MTA before the association failed:

`Octets Out =` *integer*
`MPDUs Out =` *integer*

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 The MTA aborted the association because it was unable to access one of its workspaces. This is a file access error which also generates a System Interface Error event. The System Interface Error event provides more information about the problem. Check the event sink for an occurrence of that event from this MTA and take the appropriate action as described in Section 20.1.

2 The MTA aborted the association because it was affected by an error in its own software. Check the event sink for an occurrence of the Internal Error event from this MTA and take the appropriate action as described in Section 22.2.1.

3 The MTA aborted the association because the operating system failed to do something that the MTA requested. When this error occurs, the MTA also generates the System Interface Error event. The System Interface Error event provides more information about the problem. Check the event sink for an occurrence of that event from this MTA and take the appropriate action as described in Section 20.1.

4 The MTA aborted the association because the peer MTA took too long to respond. The time limit that specifies how long the MTA can wait for a response from the peer MTA expired. Note that it is not possible to change this time limit. Contact the manager of the peer MTA and find out why the peer MTA did not respond.

5 The MTA aborted the association because it was disabled after it had established the association to the peer MTA.

Display the current state of the MTA, using the following command:

```
SHOW NODE "node-id" MTA STATE
```

If the MTA is ON, the MTA was re-enabled after the event occurred. In this case, no further action is required. If the MTA is OFF, enable the MTA using the following command:

```
ENABLE NODE "node-id" MTA
```

The MTA attempts to recover the association when you enable it.

6 The MTA aborted the association because the Peer MTA entity that holds information about the peer MTA was disabled after the MTA had established the association.

Display the current state of the Peer MTA entity, using the following command:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"]  STATE
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

If the Peer MTA entity is ON, the Peer MTA entity was re-enabled after the event occurred. In this case, no further action is required. If the Peer MTA entity is OFF, enable the Peer MTA entity using the following command:

```
ENABLE NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"]
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA.

An attempt by the MTA to recover the association can only succeed if the Peer MTA entity is in the ON state.

7 The MTA aborted the association because an MPDU that the MTA was transferring over the association was manually deleted.

8 The MTA aborted the association because the Activity entity that holds information about the association was manually deleted.

An attempt by the MTA to recover the association can fail because the recovery information held in the Activity entity has been deleted. A new association to the peer MTA will succeed. No action is required.

9 The association was aborted by one of the lower layers, for example the Transport layer, or because the physical line broke.

If this error occurs frequently, then investigate it further using the lower layer management entities. See the HP DECnet-Plus documentation for information about the lower layer management entities.

10 The MTA aborted the association because a protocol error was detected in one of its lower layers.

Check the event sink for an occurrence of the Lower Layer Protocol Violation event (see Section 16.3.2) or the RTSE Protocol Violation event (see Section 16.3.1) from this MTA. These events identify the source of the protocol error. If the information provided by these events is insufficient to identify the protocol error, then record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

11 The MTA aborted the association because it received a parameter from the peer MTA that it did not recognize. Inform the manager of the peer MTA about the problem.

To identify the parameter, record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

12 The abort is due to a problem detected by the peer MTA. In the case of the abort being due to an unrecognized activity, then no action is necessary as the initiating MTA recovers the association and re-sends the complete MPDU. If the reason is `Transfer Completed`, no further action is required. The peer MTA has already received the MPDU.

In all other cases, notify the manager of the peer MTA about the problem, and specify the Abort Reason Code. If the peer MTA is a MAILbus 400 MTA, then there should be a corresponding Inbound Failure event generated by the peer MTA. This event provides more information about why the peer MTA aborted the association.

If the failure is due to a protocol violation, it may be necessary to record the protocol information that is exchanged between the two MTAs, see Section 16.4.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

16.2.8 Transport Interface Error

This event occurs when an MTA is unable to access the Transport Service.

This could be due to one of the following:

- The Transport Service is not running. This prevents the MTA from initiating or accepting any associations with other MTAs.
- The MTA's Transport selector is being used by another application.

  Each application running on a particular node requires a unique Presentation or Session address, part of which the Transport Service uses to identify the application. The part of a Presentation or Session address used by the Transport Service is called the Transport selector (TSEL). In this case, the Transport Service is being used by another application that is using the same TSEL as the MTA. This prevents the MTA from listening for any incoming connections from peer MTAs.
- The Transport Template entity "mta_any" does not exist on the node. The MTA uses the "mta_any" Transport Template entity for inbound connections.

  This error can only occur on Tru64 UNIX systems. This error causes inbound DECnet/OSI connections to fail. Note that the MTA does not use a Transport Template entity for inbound DECnet/OSI connections when running on an OpenVMS system.

This event is counted by the MTA entity's Transport Interface Errors counter.

The event provides one of the following:

```
1  Transport Interface Error = Transport Service Unavailable
   Transport Selector = tsel
   Transport Error Code = code
   Transport Error Text = error-text

2  Transport Interface Error = Transport Selector Already in Use
   Transport Selector = tsel
   Transport Error Code = 0
   Transport Error Text = No Further Information Available

3  Transport Interface Error = Transport Template Does Not Exist
   Transport Selector = tsel
   Transport Error Code = 0
   Transport Error Text = No Further Information Available
```

where *tsel* is the Transport selector part of the MTA's Presentation or Session address, *code* is an error code provided by the Transport Service, and *error-text* is information about the error supplied by the MTA.

The error code and corresponding error text is only meaningful when the error reported by this event is `Transport Service Unavailable`.

Action

Note

Unless otherwise stated, issue the commands needed to solve the problems identified by this event at the node where the MTA that generated the event is running.

1 The error code and error text provided by the event indicates why the Transport Service is unavailable. For more information about the error code see the DECnet/OSI programming documentation.

If the MTA is using the DECnet/OSI Transport Service, the Transport Service could be unavailable because the OSI Transport entity has been deleted. If the OSI Transport entity has been deleted, when it is subsequently recreated and enabled you must also recreate the Template entities used by the MTA. See the appropriate appendix for the commands you need to run the scripts that create the MTA's Template entities.

Tru64 UNIX

On Tru64 UNIX systems, the MTA could be using the TCP/IP Transport Service. If the MTA is using the TCP/IP Transport Service, the Transport Interface Error could be due to the rfc1006 demon not running. Refer to the DECnet/OSI network documentation for information about how to check that the rfc1006 demon is running. If the demon is not running, ask the person responsible for managing the system to start it. ♦

2 Find out the MTA's Presentation address, and Session address if it has one.

Check that the MTA's Presentation Address and Session Address attributes are correct. Use the following command to display these attributes:

```
SHOW NODE "node-id" MTA PRESENTATION ADDRESS, SESSION ADDRESS
```

where *node-id* is the name of the node where the MTA that generated the event is running.

If the TSEL is incorrect, in either address, then update the MTA's address. To do this, log into a privileged account on the node where the MTA is located. Stop the MTA, and follow the instructions given in Part III of *HP MAILbus 400 MTA Planning and Setup* on how to set up an MTA and create an MTA entry in the directory. Finally, start the MTA.

If the MTA has the correct Presentation address and Session address, then check the addresses used by all the other applications running on the same node as the MTA.

Use the following command to display these addresses:

```
SHOW NODE "node-id" OSI TRANSPORT PORT * LOCAL TRANSPORT SELECTOR
```

Compare the value of each Local Transport Selector attribute displayed with the value of the MTA's Presentation Address or Session Address attributes.

Note

The Local Transport Selector attribute always displays its value in hexadecimal. However, the MTA's Presentation Address and Session Address attribute values are displayed in ASCII text if possible.

Identify the application that is using the same TSEL as the MTA. Resolve the problem with the owner of the application, so that just one application uses this TSEL value.

3 Note that this error can only occur on Tru64 UNIX systems and only effects DECnet/OSI inbound connections.

You need to create a Transport Template entity called “mta_any”. To do this, run the MTA’s CLNS Transport Template script. Refer to the appendix describing the operating system specific information for the command you need to run this script.

The MTA should start using the Template entity within a few minutes, but if you want the MTA to use it immediately, then disable and enable the MTA.

16.3 Events Relating to Protocol Violations

If an MTA sends or receives incorrect protocol information, then an association is likely to fail. Protocol violations can usually only be solved by recording the protocol information exchanged between MTAs, see Section 16.4.

When a protocol error is detected, one of the following events is generated by the MTA:

- RTSE Protocol Violation event, Section 16.3.1
- Lower Layer Protocol Violation event, Section 16.3.2

These events are always generated in conjunction with one of the following events:

- Outbound Establishment Failure
- Inbound Failure
- Outbound Failure

16.3.1 RTSE Protocol Violation

This event occurs whenever a Reliable Transfer Service Element (RTSE) generates incorrect protocol. The error can occur in the RTSE of either the MTA or the peer MTA. When an error occurs, the association is aborted by the MTA that detects the error. For example, if a peer MTA’s RTSE generates incorrect protocol that is detected by the MTA, the MTA aborts the association. This event should only occur when there is a problem interworking with an MTA that is not a MAILbus 400 MTA.

This event is counted by the RTSE Protocol Violations counter of the Peer MTA entity that holds information about the peer MTA.

The information provided by the event relates to the Reliable Transfer protocol.

If the MTA that generated the event aborted the association, then the event contains:

```
RTSE Entity Observing Protocol Violation = This MTA
Violation Detected = violation-message
Peer Event = RTSE-PDU
Local State = RTSE-state
```

where:

- *violation-message* identifies the cause of the protocol error.
- *RTSE-PDU* identifies the RTSE Protocol Data Unit (PDU) that was unacceptable to the MTA's RTSE.
- *RTSE-state* is the RTSE State Code at the MTA.

If the peer MTA aborted the association, then the event contains:

```
RTSE Entity Observing Protocol Violation = Peer MTA
Violation Detected = This MTA sent a RTSE event that the Peer MTA has
rejected
Peer Event = Not Applicable
Local State = Not Applicable
```

Action

If this event occurs when interworking, contact the manager of the peer MTA in the other routing domain. If the information provided by the event is insufficient to solve the problem, then record the protocol information that is exchanged between the two MTAs, see Section 16.4.

Note

You cannot record the protocol information that is exchanged between two MTAs that are in the same routing domain.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

If this event occurs when setting up an association between two MAILbus 400 MTAs, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

16.3.2 Lower Layer Protocol Violation

This event occurs whenever there is a protocol violation in one of the lower OSI protocol layers of either the MTA or the peer MTA. The protocol layers referred to are: Association Control Service Element (ACSE), OSI Presentation, OSI Session, or OSI Transport.

When this event occurs, the MTA that was affected by the error aborts the association. This event should only occur when there is a problem interworking with an MTA that is not a MAILbus 400 MTA.

This event is counted by the Lower Layer Protocol Violations counter of the Peer MTA entity that holds information about the peer MTA.

If the association was aborted by the MTA that generated the event, then the event contains:

```
OSI Entity = This MTA
Error Layer = OSI-layer
```

where *OSI-layer* is the name of the OSI protocol layer where the protocol error occurred.

If the association was aborted by the peer MTA, then the event contains:

```
OSI Entity = Peer MTA
Error Layer = OSI-layer
```

where *OSI-layer* is the name of the OSI protocol layer where the protocol error occurred.

Action

If this event occurs when interworking, contact the manager of the peer MTA in the other routing domain. Record the protocol information that is exchanged between the two MTAs, see Section 16.4.

Note

You cannot record the protocol information that is exchanged between two MTAs that are in the same routing domain.

If, after you have analyzed the protocol trace and consulted with the manager of the peer MTA, you are unable to correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

If this event occurs when setting up an association between two MAILbus 400 MTAs, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to provide about the error.

16.4 Recording Protocol Information

MTAs exchange protocol information every time they set up an association and every time they transfer an MPDU over the association. If an MTA that is accepting an association detects that the peer MTA is proposing to use a protocol parameter that the MTA finds unacceptable, then it either rejects or aborts the association. However, if a protocol error occurs when an association is being set up or when an MPDU is being transferred, then the association is aborted. Either the MTA that initiated the association or the peer MTA could detect the protocol error and abort the association.

Protocol errors cannot occur for associations between MTAs in your routing domain, unless there is an error in an MTA's software. However, they could occur when you are trying to set up interworking with an MTA that is not a MAILbus 400 MTA. When a protocol error occurs during interworking, you may need to record the protocol information that is exchanged over the association in order to trace the problem.

Note

You cannot record the protocol information that is exchanged between two MTAs that are in the same routing domain.

Before enabling an MTA to record protocol information it is advisable to restrict the number of associations between the two MTAs. This is because an MTA records protocol information from all the associations to or from the peer MTA. It stores the protocol information from each association in a separate file, known as a **trace binary file**. If you limit the number of associations, then you have only a few trace binary files to analyze. See Section 16.4.1 and Section 16.4.2 for information about restricting the number of associations that the MTA can initiate or receive.

The Trace attribute of the Peer MTA entity enables an MTA to record protocol information that it exchanges with a particular peer MTA in another routing domain. To create a record of the protocol information exchanged between two MTAs, set the Trace attribute in the Peer MTA entity that represents the peer MTA to ON, for example:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] TRACE ON
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain.

Note

You are advised not to permanently record protocol information, as this consumes disk space and could slow down the system that the MTA is running on. Only enable the recording of protocol information when you are trying to find out why an association to or from a particular peer MTA has failed.

After you have enabled protocol recording, the MTA records protocol information whenever an association with the peer MTA is either initiated, accepted, or recovered. The amount of protocol information that is recorded depends on whether the MTA recording the information initiated the association. If the MTA initiated the association, it records all the protocol information exchanged between the two MTAs. If the MTA receives the association, it records only the protocol information that is exchanged after it has received the connection request from the peer MTA.

After you have enabled protocol recording, monitor the association to find out if it fails. You can monitor the association by doing one of the following:

- Monitor the association using the Activity entity that provides information about the association, see (Section 16.1).

  Note that there is no Activity entity created until an association has been established and an MPDU is being transferred over the association.

- Monitoring the event sink for occurrences of the following events:
  - Outbound Establishment Failure
  - Inbound Failure
  - Outbound Failure
  - RTSE Protocol Violation
  - Lower Layer Protocol Violation
- Monitoring the counters that record occurrences of these events.

If the association fails, stop the MTA recording protocol information by setting the Trace attribute to OFF:

```
SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] TRACE OFF
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain.

Analyze the protocol information that has been recorded, see Section 16.5.

If the association does not fail, then set the Trace attribute to OFF when the transfer of the message is complete. To find out when the transfer is complete, monitor the State attribute of the related Activity entity, see Section 16.1.

16.4.1 Protocol Recording by the Initiating MTA

When an association fails, the boundary MTA that initiated the association places the MPDU that it was trying to transfer in a retry set for the peer MTA. The boundary MTA attempts to establish or recover the association when the retry interval has elapsed. See Section 7.3.5 for more information about the retry interval.

You can find out the time of the next retry attempt from the Retry Time attribute of the Peer MTA entity that represents the peer MTA. Use the following command to display this attribute:

```
SHOW NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
NAME = "peer-mta-name"] RETRY TIME
```

where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain.

If there are several MPDUs in the boundary MTA for the peer MTA, then the boundary MTA could set up more than one association to the peer MTA and create a trace binary file for each association. You can find out if the boundary MTA is likely to create more than one association using the following command:

```
SHOW NODE "node-id" MTA MPDU * TARGET
```

Check how many MPDUs have the name of the routing domain where the peer MTA is located as their target.

If there are several MPDUs for the peer MTA, then it is advisable to restrict the number of concurrent associations that the boundary MTA can initiate to the peer MTA. To do this, modify the MTA entity's Maximum Outbound Parallel Transfer Associations attribute. This attribute specifies the number of concurrent associations that the boundary MTA can have to the peer MTA.

Note

This attribute specifies the maximum number of concurrent parallel associations from the boundary MTA to each peer MTA that it is connected to. If the boundary MTA has MPDUs for transfer to several different peer MTAs, then setting this attribute to a low value could cause a delay in the transfer of MPDUs from the boundary MTA.

Before modifying this attribute, display its current value and make a note of it. Use the following command to display this attribute:

```
SHOW NODE "node-id" MTA MAXIMUM OUTBOUND PARALLEL TRANSFER ASSOCIATIONS
```

To limit the number of concurrent associations to the peer MTA, use the following command:

```
SET NODE "node-id" MTA MAXIMUM OUTBOUND PARALLEL TRANSFER -
ASSOCIATIONS 1
```

When you have finished the protocol recording, reset this attribute to its previous value.

16.4.2 Protocol Recording by the Receiving MTA

When an inbound association fails, the recovery of the association is the responsibility of the initiating peer MTA in the other routing domain. Contact the manager of the peer MTA and find out how the peer MTA is likely to respond. If the peer MTA is a MAILbus 400 MTA, then it tries to recover the association.

The peer MTA may initiate more than one association to the boundary MTA that is recording the protocol information. The boundary MTA produces a trace binary file for each association from the peer MTA. To reduce the number of binary trace files that you have to analyze, ask the manager of the peer MTA to restrict the number of concurrent associations from the peer MTA. If the peer MTA is a MAILbus 400 MTA, then the peer MTA manager can do this by modifying the Maximum Outbound Parallel Transfer Associations attribute, as described in Section 16.4.1.

If it is not possible for the manager of the peer MTA to restrict outbound associations from the peer MTA, then restrict the number of inbound associations that the boundary MTA can receive concurrently. The number of concurrent inbound associations is specified by the MTA entity’s Maximum Inbound Transfer Associations attribute.

Note

If the boundary MTA is connected to several peer MTAs, then setting this attribute to a low value prevents the boundary MTA accepting associations from other peer MTAs. This could cause congestion in your routing domain.

Before modifying this attribute, display its current value and make a note of it. Use the following command to display this attribute:

```
SHOW NODE "node-id" MTA MAXIMUM INBOUND TRANSFER ASSOCIATIONS
```

Use the following command to set this attribute:

```
SET NODE "node-id" MTA MAXIMUM INBOUND TRANSFER ASSOCIATIONS value
```

where *value* is the maximum number of associations that you want the MTA to be able to accept.

When you have finished the protocol recording, reset this attribute to its previous value.

16.5 Analyzing the Protocol Trace File

The boundary MTA records protocol information in binary format. The information is recorded sequentially as an MPDU is transferred over an association. The boundary MTA stores the protocol information it records in a trace binary file. The boundary MTA creates one trace binary file for each association to or from the peer MTA during the time that protocol recording is enabled.

The boundary MTA places each trace binary file it creates in its trace directory. For the location of the boundary MTA's trace directory, refer to the appendix describing the operating system specific information.

The boundary MTA assigns a unique name to each trace binary file it creates. The name of the file depends on whether the boundary MTA that created the file initiated or received the association:

- *name*_INIT_****_****

  This filename format is used when the boundary MTA that recorded the protocol information initiated the association.

- *name*_RCV_****_****

  This filename format is used when the boundary MTA that recorded the protocol information received the association from the peer MTA.

where *name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain, and `****_****` represents a timestamp and some digits derived from the processes identifier. These make the filename unique.

OpenVMS

On OpenVMS systems, the names of trace binary files are terminated with a period (.).
♦

An MTA is supplied with a Trace Analyzer utility, this is software that you invoke from the operating system prompt. The Trace Analyzer analyzes a trace binary file and converts the protocol information from binary to text.

You must run the Trace Analyzer from a privileged account. For information about the privileges that you need, refer to the appendix describing the operating system specific information.

To run the Trace Analyzer, log into a privileged account on the node where the MTA is running and issue the following command at your operating system prompt:

Tru64 UNIX

```
# osaktrace [options] in_file
```

where:

- ***options*** is one or more of the following qualifiers:

-t	Trace Transport events only
-h	Trace Transport events without converting contents of the Transport Service Data Unit (TSDU)
-s	Trace Session Layer protocols only
-p	Trace Presentation Layer protocols only
-a	Trace ACSE protocols only
-u	Trace user data only
-e	Find errors
-d	Convert Define Context Set (DCS) setup and verification information from binary to text
-f	Replace time and date strings with `"XXX"`. For example, replace `15:24:59.94` on `14-MAY-1991` with `XX:XX:XX.XX` on `XX-XXX-XXXX`
-o	Send the output to a named file

- ***in_file*** is the name of the trace binary file that you want to analyze.

If you specify the **-o** option, then supply the name of the output file immediately after the **-o** option qualifier. If you do not specify an output file, then the output is displayed on the screen. If you do not specify any options, the Trace Analyzer uses a default set of options as follows:

```
-t -s -p -a
```

If you select the errors option, the Trace Analyzer attempts to detect protocol errors. This option is most likely to be useful when you are tracing Presentation layer and ACSE protocol information. ♦

OpenVMS

$ **OSAKTRACE [*options*] *in_file***

where:

- ***options*** is one or more of the following qualifiers:

/TRANSPORT_EVENTS	Trace Transport events only
/NOTRANSPORT_EVENTS	Do not trace Transport events
/SESSION_PCI	Trace Session Layer protocols only
/NOSESSION_PCI	Do not trace Session Layer protocols
/PRESENTATION_PCI	Trace Presentation Layer protocols only
/NOPRESENTATION_PCI	Do not trace Presentation Layer protocols
/ACSE_PCI	Trace ACSE protocols only
/NOACSE_PCI	Do not trace ACSE protocols
/USER_DATA	Trace user data only
/NOUSER_DATA	Do not trace user data
/ERRORS	Find errors
/NOERRORS	Do not find errors
/DCS	Convert Define Context Set (DCS) setup and verification information from binary to text
/NODCS	Do not convert Define Context Set (DCS)
/HEADERS_ONLY	Output the Transport event headers only. This qualifier overrides any of the previous qualifiers

/FILTER	Replace time and date strings with `"XXX"`. For example, replace `15:24:59.94` on `14-MAY-1991` with `XX:XX:XX.XX` on `XX-XXX-XXXX`
/NOFILTER	Do not replace time and date strings with `"XXX"`
/OUTPUT	Send the output to a named file

- ***in_file*** is the name of the trace binary file that you want to analyze.

If you specify the **/OUTPUT** qualifier, then supply the name of the output file immediately after the **/OUTPUT** qualifier, for example:

```
$ OSAKTRACE/OUTPUT = my_output_trace_file.txt in_file
```

If you do not specify an output file, then the output is written to the device defined by SYS$OUTPUT.

If you do not specify any options, the Trace Analyzer uses the following qualifiers:

```
/TRANSPORT_EVENTS
/SESSION_PCI
/PRESENTATION_PCI
/ACSE_PCI
/NOUSER_DATA
/NOFILTER
```

If you select the **/ERRORS** qualifier, the Trace Analyzer attempts to detect protocol errors. This qualifier is most likely to be useful when you are tracing Presentation layer and ACSE protocol information.
♦

Check through the readable protocol trace and identify the protocol error.

If, after you have checked the output from the Trace Analyzer you are unable to identify or correct the error, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to supply about the problem.

17 How the MTA Makes Connections Over X.25

This chapter describes some of the management entities and attributes used within OSI Transport and X.25. This information explains how MTA inbound and outbound connections relate to the entities and attributes in the OSI Transport module and X.25 modules.

You can use this information to try and identify problems when X.25 connections fail. The information described in this chapter assumes that OSI Transport and X.25 have been installed and configured as described in the documentation related to each of the products. No information about configuring X.25 or OSI Transport is provided in this chapter. The chapter also assumes that you have set up your MTA as described in Part III of *HP MAILbus 400 MTA Planning and Setup*.

The chapter is intended to help isolate where and why problems occur, it does not provide instructions for solving problems related to OSI Transport or X.25. Refer to the documentation related to each of the products for specific trouble shooting information.

The chapter does not describe the use of LLC2 or X.25 Relay. The chapter describes how the MTA uses the X.25 Native and X.25 Gateway Client capabilities of X.25.

Information about how to customize the MTA's use of the Transport Service is provided in Section 7.7.

In particular, this chapter describes:

- The management entities that need to exist for the MTA to be able to make connections and how these entities relate to each other (Section 17.1).
- Potential problems with X.25 impacting the MTA (Section 17.2).
- Isolating failed X.25 connections (Section 17.3).

Within this chapter:

- An X.25 Server or X.25 Native system is a system connected directly to one or more Packet Switching Data Networks (PSDNs) and conforms to CCITT Recommendation X.25.

  The X.25 Server capability is currently available only on OpenVMS VAX.

  Your PTT is responsible for providing access to the PSDN; the PTT also allocates Data Terminal Equipment (DTE) addresses. It is this DTE address that is used when contacting another system using X.25.
- An X.25 Gateway Client is a system that is not connected directly to the PSDN. An X.25 Gateway Client accesses a PSDN through an X.25 Server. The client connects to the server using DECnet/OSI.

  The PTT allocates Data Terminal Equipment (DTE) addresses for the server. It is this address, plus a subaddress, that is used when the client contacts another system using X.25.

  The X.25 Gateway Client capability is currently only available on OpenVMS and ULTRIX™.
- Call Data Value (CDV) is a characteristic attribute of the X25 Access Filter entity. However, within the protocols CDV is defined as Call User Data (CUD). Throughout this chapter CDV is used to describe the management attribute and the protocol.

In addition to the information provided in the MAILbus 400 MTA documentation, you will need to have available the DECnet/OSI documentation, X.25 documentation and Common Trace Facility documentation.

17.1 Management Entities

The following sections describe the management entities that hold information for inbound and outbound connections to and from peer systems using X.25.

Section 17.1.1 describes the management entities required in order that the MAILbus 400 MTA can make outbound connection requests when installed on an X.25 Server or X.25 Native system.

Section 17.1.2 describes the additional management entities required at an X.25 Server in order that it can make outbound connection requests on behalf of its clients.

Section 17.1.3 describes the management entities required in order that the MAILbus 400 MTA can make outbound connection requests when installed on an X.25 Gateway Client.

Section 17.1.4 describes the management entities required in order that the MAILbus 400 MTA can accept inbound connection requests when installed on an X.25 Server or X.25 Native system.

Section 17.1.5 describes the management entities required at an X.25 Server in order that it can accept inbound connection requests on behalf of its clients.

Section 17.1.6 describes the management entities required in order that the MAILbus 400 MTA can listen for inbound connection requests.

17.1.1 Outbound Connection Requests at an X.25 Server or X.25 Native System

Figure 17–1 shows the relationship between the different entities when the MAILbus 400 MTA attempts to make an outbound connection. The callouts in Figure 17–1 are explained in the following list:

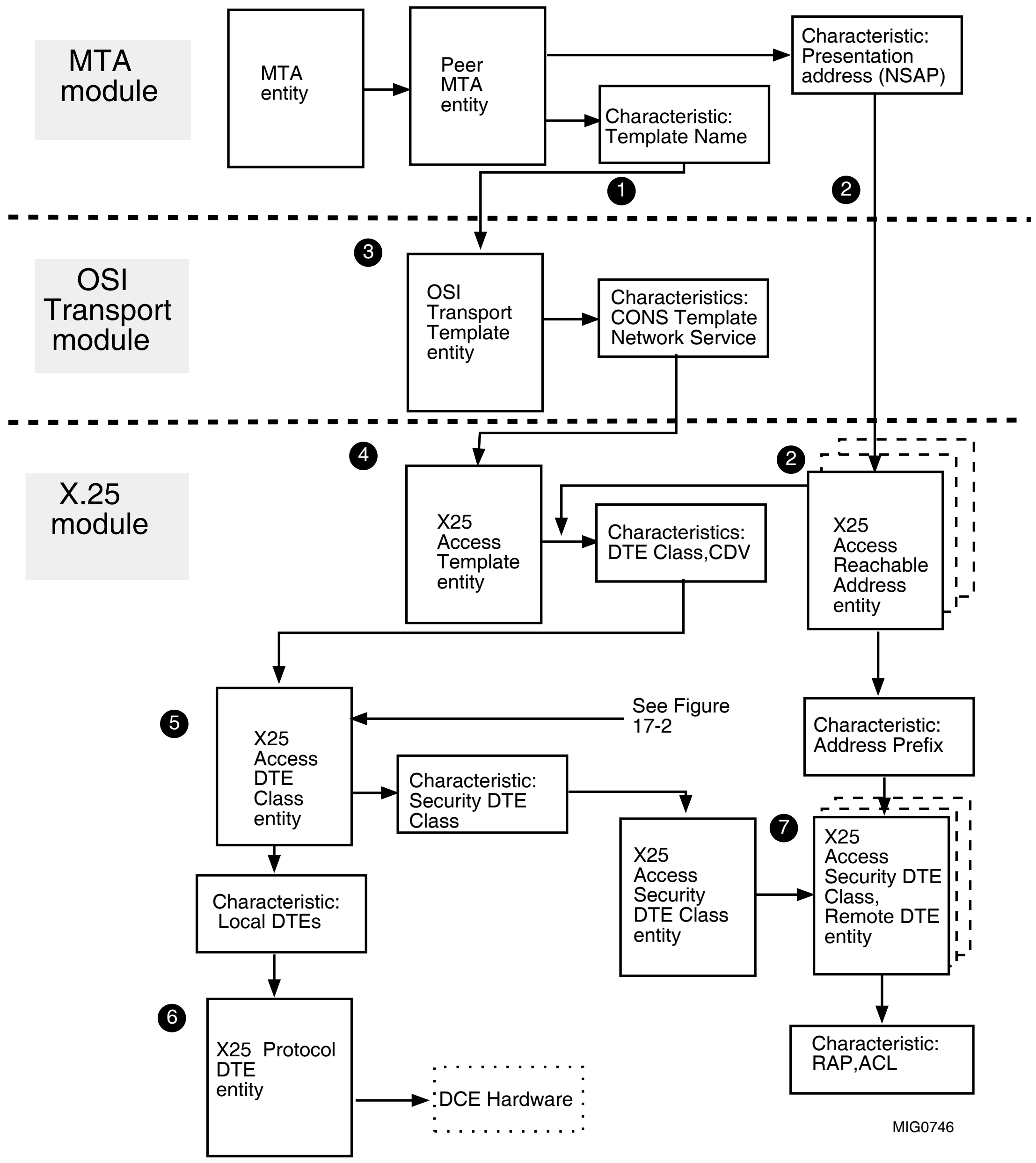
1 The Peer MTA entity, an entity of the MTA module, has a Template Name attribute. This attribute identifies one or more OSI Transport Template entities that are used by the OSI Transport Service for outbound connections to peer MTAs. For CONS connections the OSI Transport Template must include the "mta_cons" template, or an equivalent template with the Network Service attribute set to CONS.

2 OSI Transport expects a CONS NSAP for the target to make CONS connections. This CONS NSAP is taken from the Presentation or Session Address attribute in the Peer MTA entity; see Section 7.7.

X.25 expects a DTE address and therefore OSI Transport invokes the conversion of the CONS NSAP to a DTE address through the X25 Access Reachable Address entity. An attribute of the X25 Access Reachable Address entity is Address Prefix. The Address Prefix attribute identifies the X25 Reachable Address entity that most closely matches the CONS NSAP provided.

The Address Prefix attribute can be a complete or partial CONS NSAP. A partial CONS NSAP is represented by one X25 Access Reachable Address entity. If the Address Prefix attribute is a partial CONS NSAP, it must have the attribute Mapping set to "X.121". You should also check the value for the Address Extensions attribute in the X25 Access Reachable Address entity with your PSDN provider, as this value is specific to individual PSDNs.

Figure 17–1 Entities Used for Outbound Connections


MTA module
MTA entity
Peer MTA entity
Characteristic: Presentation address (NSAP)
Characteristic: Template Name
1
2
OSI Transport module
3
OSI Transport Template entity
Characteristics: CONS Template Network Service
X.25 module
4
X25 Access Template entity
Characteristics: DTE Class,CDV
2
X25 Access Reachable Address entity
5
X25 Access DTE Class entity
See Figure 17-2
Characteristic: Address Prefix
Characteristic: Security DTE Class
X25 Access Security DTE Class entity
7
X25 Access Security DTE Class, Remote DTE entity
Characteristic: Local DTEs
6
X25 Protocol DTE entity
DCE Hardware
Characteristic: RAP,ACL
MIG0746

Note

The system making the call must have an OSI Transport CONS NSAP Addresses attribute that contains at least one valid CONS NSAP address before an X.25 outbound call can be made.

---

3 The OSI Transport Template entity is an entity of the OSI Transport module. One of the attributes of the OSI Transport Template is CONS Template, which identifies the X.25 Access Template that the MTA will use for outbound connections. Another attribute of the OSI Transport Template is Network Service, which must be set to CONS. The MTA's OSI Transport Template and characteristics are created when the MTA is set up; see Part III of *HP MAILbus 400 MTA Planning and Setup*.

You should also check the value for the Expedited Data attribute in the OSI Transport Template with your PSDN provider, as this value is specific to individual PSDNs.

4 The X25 Access Template is an entity of the X25 Access module. Once you have identified the X25 Access Template, using the CONS Template value from step 3, you can find out the DTE Class and Call Data Value (CDV) specific to OSI Transport.

The specific value required by OSI Transport is:

```
SET X25 ACCESS TEMPLATE "temp-name" CALL DATA = '03010100'H
```

The DTE Class is an attribute of the X25 Access Template entity. The attribute DTE Class is also an attribute of the X25 Access Reachable Address entity; see step 2. The DTE Class attribute identifies an X25 Access DTE Class entity.

5 The DTE Class attribute identified from the X25 Access Template entity or the X25 Access Reachable Address entity in step 4 identifies the X25 Access DTE Class entity. One of the attributes of this entity is the Security DTE Class, which controls inbound and outbound access (see step 7), another attribute is Local DTEs (see step 6).

6 The Local DTEs characteristic identifies an X25 Protocol DTE entity that defines the Data Circuit-terminating Equipment (DCE) hardware to use for the outbound call.

7 The Security DTE Class identifies an X25 Access Security DTE Class entity (see step 5), which has a subentity Remote DTE.

There can be more than one Remote DTE entity for each X25 Access Security DTE Class. The Remote DTE entity with a Remote Address Prefix (RAP) attribute that most closely matches the DTE address identified in step 2 is selected. The RAP attribute might be a complete or partial DTE address.

The selected Remote DTE entity has an ACL attribute that provides details of whether the caller has the necessary authority to make the outbound call to the peer system, including the security level allowed for each Rights Identifier value.

In the case of an X.25 Native or X.25 Server system, the process that initiates the call has one or more Rights Identifiers; for example, the MAILbus 400 MTA process.

Tru64 UNIX

On Tru64 UNIX systems, the MTA has root privileges and Rights Identifiers. Rights Identifiers are equivalent to Groups, which are defined in /etc/group. Any Group that includes root also includes the MTA.

♦

OpenVMS

On OpenVMS systems, Rights Identifiers are created using the Authorize utility. Rights Identifiers are granted to a particular user. The MTA process is started by System, therefore the MTA inherits the System Rights Identifiers.

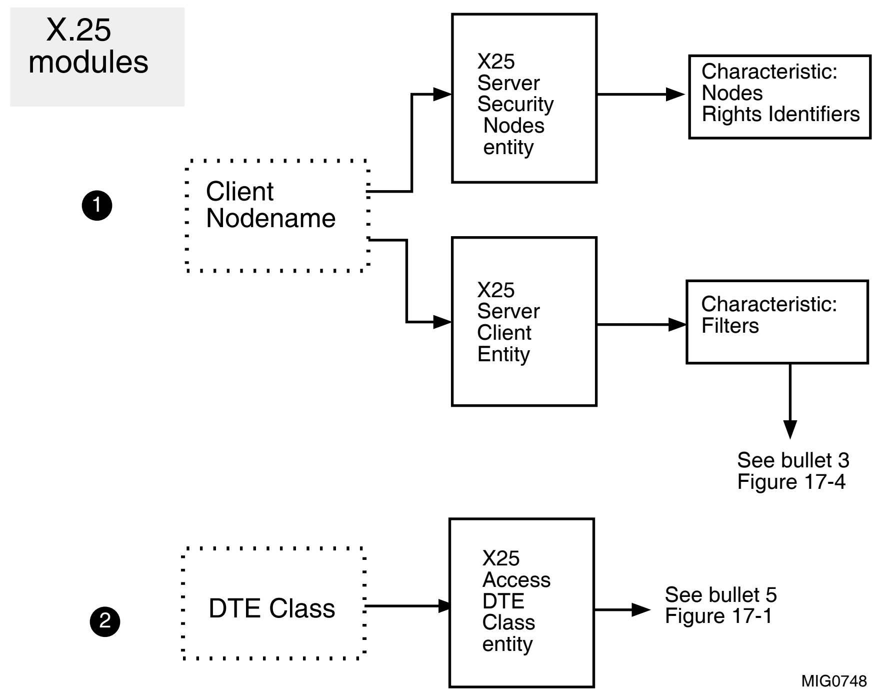
♦

In the case of an X.25 Client system, the Rights Identifier is obtained from the corresponding X25 Server Security Nodes entity (see step 1 in Section 17.1.2).

17.1.2 Additional X.25 Management Entities Required for an X.25 Server, Outbound

The entities and attributes described in this section are required on the X.25 Server system when making outbound calls on behalf of an X.25 Gateway Client.

Figure 17–2 X.25 Entities Required at X.25 Server for Outbound Connections



The callouts on Figure 17–2 are explained in the following list:

1 Before attempting to make the connection to the peer system, the X.25 Server needs to check that the X.25 Gateway Client is authorized to make the outbound connection. The X.25 Server does this by completing an inbound security check as described in Section 17.1.4 steps 1 to 4.

On the X.25 Server system, the X25 Server Security Nodes entity provides the security information required for the X.25 Gateway Client. The X25 Server Security Nodes entity has an attribute Rights Identifier, which is used with the ACL attribute of the X25 Access Security Filter entity.

If the inbound security check is successful, an X25 Server Client entity is selected. The X25 Server Client entity has an attribute Filters that provides the identity of an X25 Access Filter entity (see step 3 in Section 17.1.4). The X25 Access Filter entity has an attribute Security Filter, which provides the identity of an X25 Access Security Filter entity. The X25 Access Security Filter entity has an attribute ACL, which provides the access level permitted for the Rights Identifier identified in the X25 Server Security Nodes entity.

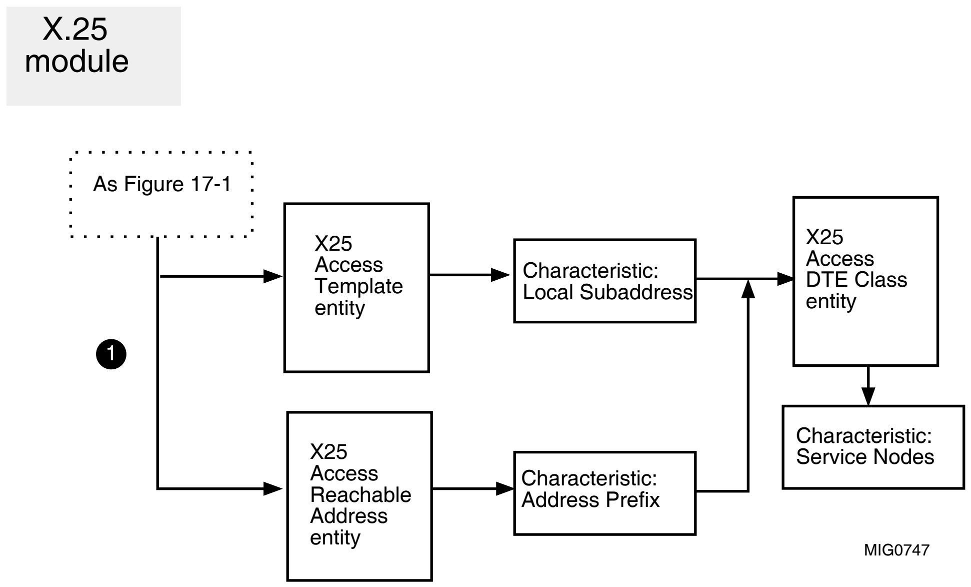
2 The X.25 Gateway Client passes the DTE Class name to the X.25 Server. This DTE Class must exist on both the X.25 Server and the X.25 Gateway Client. On the X.25 Server the Type attribute has a value “Local DTE”, and on the X.25 Gateway Client, the Type attribute has the value “remote”. The X.25 Server uses the DTE Class to make the outbound call.

The X.25 Server then completes the outbound connection request as described in Section 17.1.1, step 5 onwards.

17.1.3 Additional X.25 Management Entities Required for an X.25 Gateway Client, Outbound

Figure 17–3 shows the additional entities that are used between X.25 Gateway Clients and their X.25 Servers when making outbound calls, after steps 1 to 3, described in Section 17.1.1, are completed successfully.

Figure 17–3 X.25 Entities Required for an X.25 Gateway Client for Outbound Connections


X.25 module
As Figure 17-1
1
X25 Access Template entity
Characteristic: Local Subaddress
X25 Access DTE Class entity
Characteristic: Service Nodes
X25 Access Reachable Address entity
Characteristic: Address Prefix
MIG0747

This section describes the entities and attributes that are required on the X.25 Gateway Client, so the client can forward the outbound connection request to the X.25 Server.

The following is an explanation of the callout in Figure 17–3:

1 At the node where the X.25 Gateway Client is installed, the X25 Access Template entity provides the Local Subaddress attribute (DTE address suffix of the X.25 Gateway Client). The value of the Local Subaddress should be one of the values specified in the X.25 Access Filter (Subaddress Range attribute), for the X.25 Client, on the X.25 Server system.

The DTE Class is an attribute of the X25 Access Template entity. The attribute DTE Class is also an attribute of X25 Access Reachable Address entity. The attribute DTE class identifies an X25 Access DTE Class entity. The X25 Access DTE Class entity must exist on both the X.25 Gateway Client and the X.25 Server. For the X.25 Gateway Client, the Type attribute of the X25 Access DTE Class entity has the value “Remote”, which indicates that the DTE Class is for an X.25 Gateway Client system, and the Service Nodes attribute value identifies the X.25 Server system for the X.25 Gateway Client. For the X.25 Server the Type attribute of the X25 Access DTE Class entity has the value “Local DTE”.

The X.25 Gateway Client completes the security check as described in Section 17.1.1 step 7, and then the DTE Class, Local Subaddress and Nodename attribute values are passed in the protocol between the X.25 Gateway Client and the X.25 Server. The attributes are used by the X.25 Server to make the outbound connection request as described in Section 17.1.1, step 5 onwards.

Note

Be aware that if, as an X.25 Gateway Client system, you have more than one X.25 Server system, and you have different subaddresses on each server system, you will need an X25 Access Template entity for each system. The Local Subaddress attribute of the X25 Access Template is set according to the different subaddresses assigned on each X.25 Server system. You also need an OSI Transport Template entity for each X25 Access Template entity that you set up.

17.1.4 Management of Inbound Connections at an X.25 Server or Native System

Figure 17–4 shows the relationship between the different entities when an inbound connection request is made to the MAILbus 400 MTA. The callouts on Figure 17–4 are explained in the following list:

1 When an inbound connection is made from a peer system, X.25 takes the DTE address provided by the caller and searches all the Remote DTE entities, (subentities of the X25 Security DTE Class entity) for a Remote Address Prefix (RAP) attribute that most closely matches the DTE address provided. The RAP can be a complete or partial DTE address.

  The DTE address provided can be a DTE address of an X.25 Gateway Client or the DTE address received directly from the PSDN.

  Once the match is made, the Rights Identifier(s) associated with the RAP is used for authentication (see step 3).

2 The caller also provides a number of parameters. The X25 Access Filter entity is identified on the basis of, and in the order of:

  1. Priority (provided through the management entity)
  2. CDV (provided by the caller)

     The match on CDV is done using the CDV and Call Data Mask (CDM) attributes. A null CDV in the X25 Access Filter entity means match on anything.

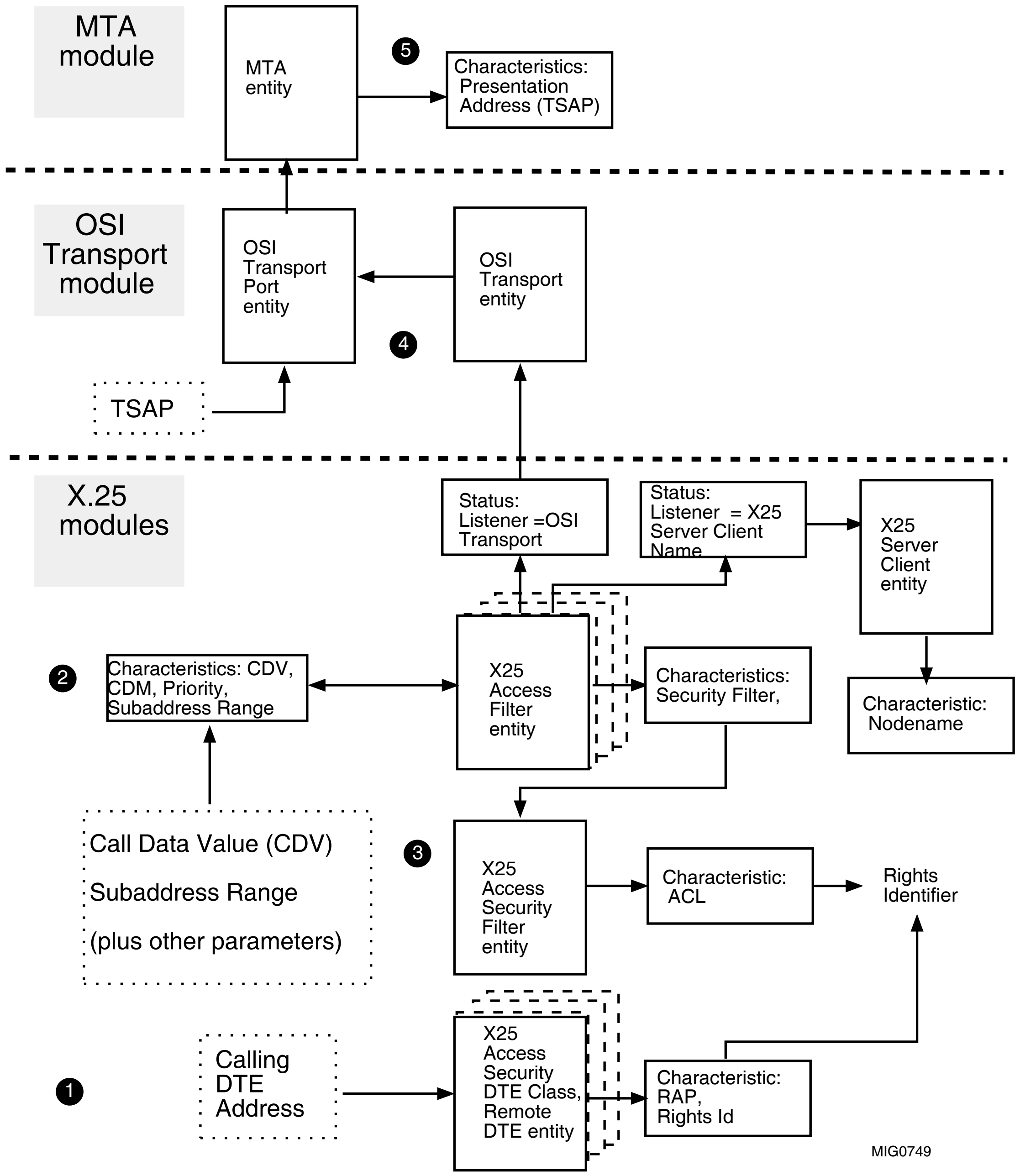
     OSI Transport requires specific values for these parameters, as shown in the following example:

     ```
     SET X25 ACCESS FILTER "temp-name" CALL DATA VALUE='03010100'H, -
      CALL DATA MASK='ffffffff'H
     ```

     Note that the Identifier attribute of the X25 Access Filter entity must match the Name Identifier attribute of the X25 Access Template entity specified in Section 17.1.1, step 4.

  3. Subaddress Range (optional and provided by the caller)
  4. Any other parameters provided in the call that are defined in the X.25 Access Filter.

Figure 17–4 Entities Used for Inbound Connections at an X.25 Server or X.25 Native System


MTA module
MTA entity
5
Characteristics: Presentation Address (TSAP)
OSI Transport module
OSI Transport Port entity
OSI Transport entity
4
TSAP
X.25 modules
Status: Listener =OSI Transport
Status: Listener = X25 Server Client Name
X25 Server Client entity
2
Characteristics: CDV, CDM, Priority, Subaddress Range
X25 Access Filter entity
Characteristics: Security Filter,
Characteristic: Nodename
Call Data Value (CDV)
Subaddress Range
(plus other parameters)
3
X25 Access Security Filter entity
Characteristic: ACL
Rights Identifier
1
Calling DTE Address
X25 Access Security DTE Class, Remote DTE entity
Characteristic: RAP, Rights Id
MIG0749

Ensure that the X25 Access Filter identified is associated with a listener and that the filter status is “In Use”. For example, the Status attribute Listener is set to OSI Transport (for an X.25 Server or X.25 Native system), or the name of a X25 Server Client entity (for an X.25 Gateway Client).

3 The Security Filter attribute of the X25 Access Filter identifies an X25 Access Security Filter entity. An attribute of the X25 Security Access Filter entity is ACL. The Rights Identifier from step 1 is used to determine if the connection can be accepted. The Identifier in the ACL must match the Rights Identifier, and the Access Level must be set in order that the call can be accepted. If the Access Level is None, the call is not accepted. If the Rights Identifier does not exist in the ACL the Access Level is taken as None, and the call is rejected by X.25. The connection is passed to OSI Transport, or the X.25 Server Client entity, if the validation is successful.

4 When the X.25 connection is validated, the inbound call is passed to the Listener identified in step 2.

You need to make sure that the OSI Transport entity has an attribute CONS NSAP Addresses, which contains a list of one or more CONS NSAPs for your system.

Tru64 UNIX

On Tru64 UNIX systems, OSI Transport compares the called CONS NSAP with the addresses specified in the CONS NSAP Addresses attribute. If the CONS NSAP for the call does not match one of the addresses in this attribute, the connection is rejected by OSI Transport.

♦

OpenVMS

On OpenVMS systems, the Inbound attribute of the OSI Transport Template entity must be set to “True” for inbound connections. In addition, the name of the X.25 Access Template must be the same as the name of the X.25 Access Filter for the call. Be aware that the template name is case sensitive.

♦

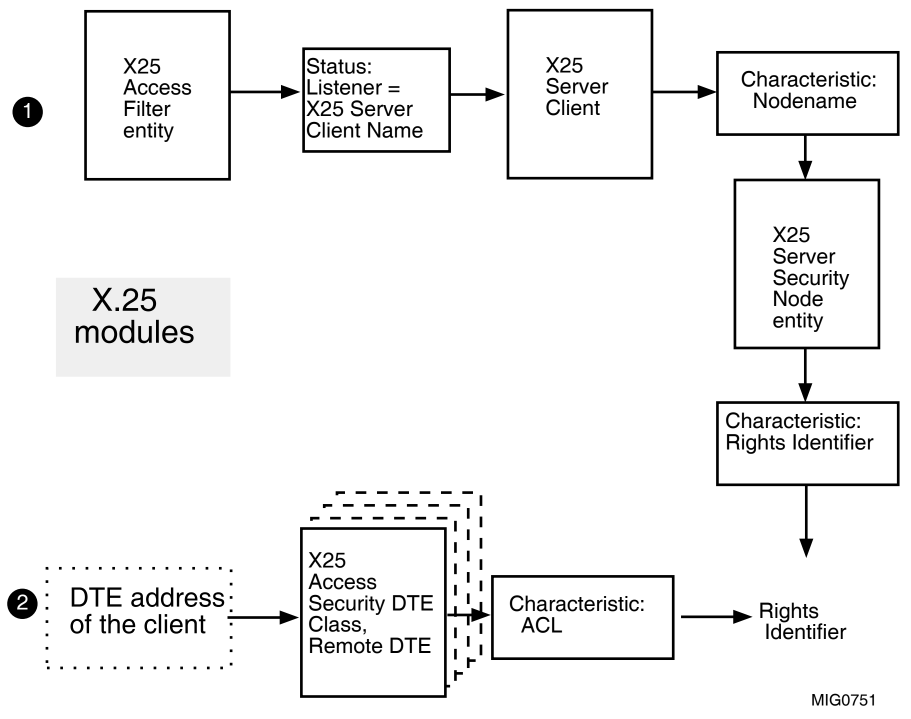
5 OSI Transport compares the TSAP in the called address with the value in the MTA’s Presentation address, and if it matches, the connection request is then passed to the MAILbus 400 MTA.

If the comparison fails, OSI Transport rejects the connection. The connection is also rejected if the MTA is not listening for incoming connections from OSI Transport. Section 17.1.6 describes how to check whether or not the MTA is listening for incoming connections.

17.1.5 X.25 Entities Required at an X.25 Server for an X.25 Gateway Client, Inbound

Figure 17–5 shows the additional entities that are provided at X.25 Servers in order that the server can accept inbound calls on behalf of its clients. The X.25 Server must verify that the X.25 Gateway Client is authorized to accept the inbound connection request.

Figure 17–5 X.25 Server Entities for Inbound Connections to an X.25 Gateway Client


1
X25 Access Filter entity
Status: Listener = X25 Server Client Name
X25 Server Client
Characteristic: Nodename
X25 Server Security Node entity
Characteristic: Rights Identifier
X.25 modules
2
DTE address of the client
X25 Access Security DTE Class, Remote DTE
Characteristic: ACL
Rights Identifier
MIG0751

The callouts on Figure 17–5 are explained in the following list:

1 An X25 Server Client entity describes one client that a particular X.25 Server can contact. The Nodename attribute is the DNS name of the client. See Section 17.1.4 step 2.

The Nodename attribute is used to identify an X25 Server Security Nodes entity, which has a characteristic attribute Rights Identifiers.

2 When an inbound connection is made, X.25 takes the DTE address provided by the caller and searches all the Remote DTE entities (subentities of the X25 Security DTE Class entity) for a RAP attribute that most closely matches the DTE address provided. There can be more than one Remote DTE entity for each X25 Access Security DTE Class. The Remote DTE entity with a Remote Address Prefix (RAP) attribute that most closely matches the DTE address identified is selected. The RAP may contain a complete or partial DTE address.

   The Rights Identifier from step 1 is used to determine if the connection can be accepted. The Identifier in the ACL must match the Rights Identifier, and the Access Level must be set in order that the call can be accepted. If the Access Level is None, the call is not accepted. If the Rights Identifier does not exist in the ACL the Access Level is taken as None, and the call is rejected by X.25.

   The X.25 Server then forwards the inbound connection request to the X.25 Gateway Client as described in Section 17.1.1 step 7.

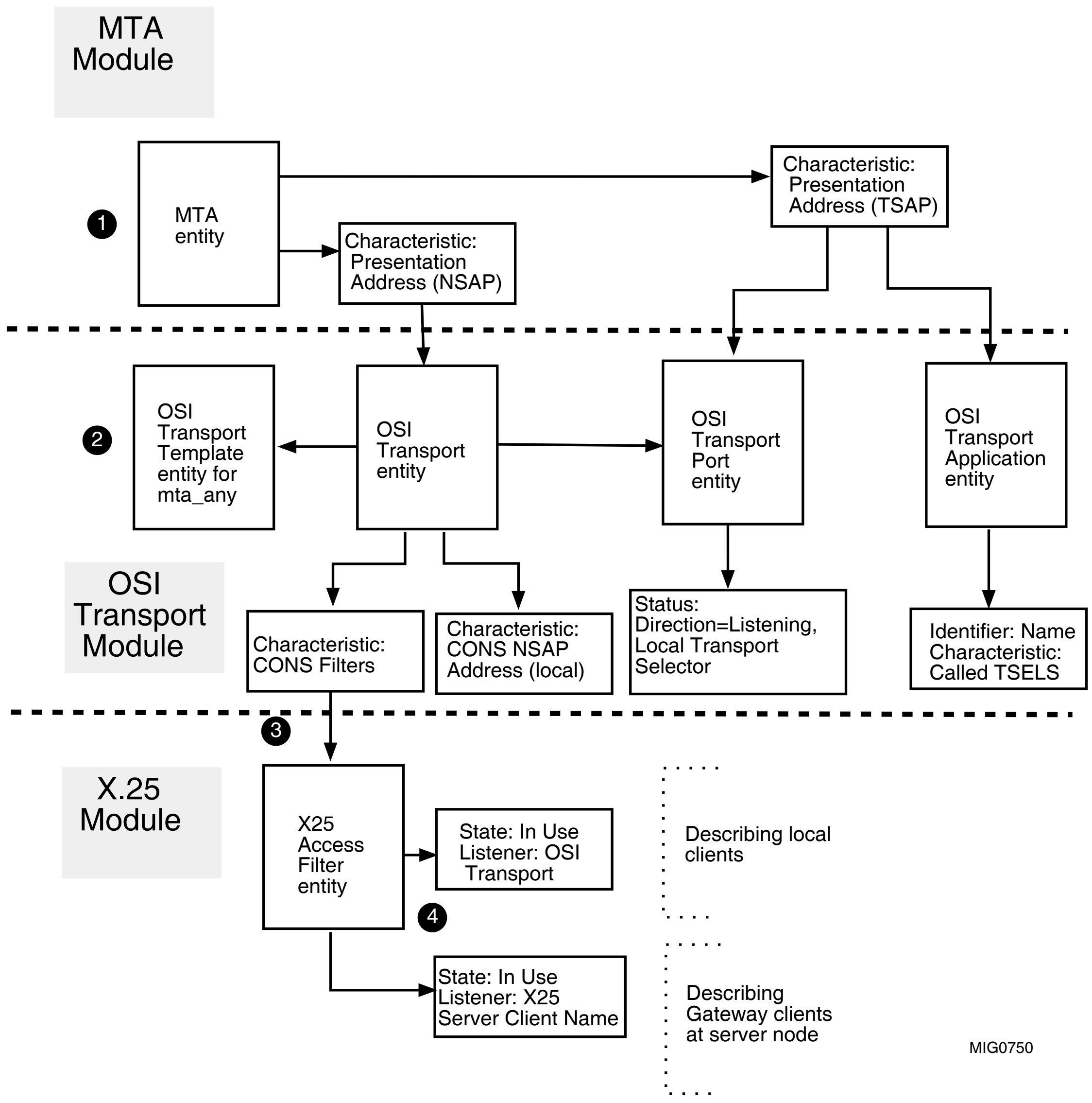
17.1.6 How the MTA Listens for Incoming Connections

In order for an MTA to be able to accept an incoming connection, it must be able to listen for the connection. Figure 17–6 shows the entities and attributes that are required for the MTA to listen for connections. The callouts on Figure 17–6 are explained in the following list:

1 The value of the Presentation address or Session Address attribute of the MTA entity contains a Transport Selector (TSAP) and a Network Service Access Point (NSAP).

2 One of the NSAPs in the MTA’s Presentation address or Session address needs to match one of the NSAPs in the CONS NSAP Address attribute of the OSI Transport entity (see step 4 in Section 17.1.4).

Figure 17–6 Entities Used When Listening for Connections


MTA Module
1
MTA entity
Characteristic: Presentation Address (TSAP)
Characteristic: Presentation Address (NSAP)
2
OSI Transport Template entity for mta_any
OSI Transport entity
OSI Transport Port entity
OSI Transport Application entity
OSI Transport Module
Characteristic: CONS Filters
Characteristic: CONS NSAP Address (local)
Status: Direction=Listening, Local Transport Selector
Identifier: Name Characteristic: Called TSELS
3
X.25 Module
X25 Access Filter entity
State: In Use Listener: OSI Transport
4
Describing local clients
State: In Use Listener: X25 Server Client Name
Describing Gateway clients at server node
MIG0750

Tru64 UNIX

On Tru64 UNIX systems, when the MTA is enabled and is listening, an OSI Transport Port entity is created with a status attribute Local Transport Selector that contains the hexadecimal value of the MTA’s TSAP. The OSI Transport Port entity also has a status attribute Direction with a value of “Listening”.

The MTA always uses the “mta_any” Transport Template for inbound communications (see Section 7.7). OSI Transport rejects the connection if the OSI Transport Template entity “mta_any” contains a Network Service attribute that does not include CONS or ANY. OSI Transport also rejects the connection if the incoming transport class does not include those defined in the OSI Transport Template entity “mta_any”.

♦

OpenVMS

On OpenVMS systems, when the MTA is enabled and is listening, an OSI Transport Application entity is created with an identifier attribute Name that contains the value of the MTA’s TSAP.

The OSI Transport Template entity “mta_cons” has the attribute Inbound set to “True”, and the Network Service attribute set to “CONS”. The mta_cons OSI Transport Template can therefore be used for inbound connections if there is no other suitable template already defined. OSI Transport rejects the inbound connection if the incoming transport classes do not include those classes defined in the OSI Transport Template entity being used for the inbound call.

♦

3 When the OSI Transport entity is enabled and has a CONS Filters attribute set to identify an X25 Access Filter entity, the X25 Access Filter entity’s Status attributes are automatically set to Direction “In Use”, and Listener is “OSI Transport”. If the Status attribute of the X25 Access Filter is “Free” the filter is not being used by any listening processes.

OpenVMS

On OpenVMS systems, the name of the X25 Access Filter entity must be the same as the name of the X25 Access Template entity that exists on the system.

♦

4 If the node is an X.25 Gateway Client, when the X25 Server Client entity is enabled at the X.25 Server system, the X25 Access Filter entity’s Status attribute on the X.25 Server system is “In Use” and its Listener is identified as the name of the X.25 Gateway Client that it serves. If the status of the X25 Access Filter is “Free” the filter is not being used by any listening processes.

17.2 Problems Related to X.25 That Affect the MTA

The setup procedure for X.25 keeps a data file for any commands entered using the setup procedures. Each time the X.25 setup procedure is executed, this data file is read to obtain the current setup information. The data file is used in preference to the NCL scripts that you might also have updated. Be aware that if you modify X.25 NCL scripts, and subsequently execute the setup procedure, the modified NCL scripts are overwritten during setup.

You must decide whether to always use the X.25 setup procedure or to edit the X.25 NCL scripts directly.

The following are problems that might exist on your system after you have installed and configured X.25 Native:

- Incorrect X25 Access Reachable Address entity.

  Check that the X25 Access Reachable Address entity is set up correctly. The following is an example of the correct command to set up an X25 Access Reachable Address entity:

  ```
  SET X25 ACCESS REACHABLE ADDRESS name DTE CLASS dte_class
  ```

  where *name* is the name provided for the entity and *dte_class* is the DTE Class attribute value for the entity. The DTE Class attribute is either specified in the X25 Access Reachable Address entity, or the X.25 Access Template entity.

  See step 4 in Section 17.1.1.

- X.29 Login not working.

  Check that X.29 Login is working; this will confirm that it is possible to make an X.25 connection when only parts of the system are configured correctly.

17.3 Isolating Failed Connections

This section describes what you can do if your system is failing to make X.25 connections.

You need to check outbound connections from your system as described in Section 17.3.1 and inbound connections to the target system as described in Section 17.3.2.

17.3.1 Failed Outbound Connections

Start by checking whether the connect request is leaving your system by tracing the connection as described in the Common Trace Facility documentation.

If the trace on the calling system indicates that the connect request is being sent by your system, check the inbound connection on the target system as described in Section 17.3.2.

If the trace on the target system indicates that the connect request is not being sent by your system, investigate the following:

- Is the CONS Template attribute of the OSI Transport Template entity correct?

  Do this by checking that there is an X25 Access Template with a name that matches the CONS Template attribute name of the OSI Transport Template, as follows:

  ```
  SHOW OSI TRANSPORT TEMPLATE temp-name CONS TEMPLATE
  SHOW X25 ACCESS TEMPLATE name ALL
  ```

  where ***temp-name*** is the name of the OSI Transport Template entity used by your application, for example mta_cons, and ***name*** is the value of the CONS Template attribute for ***temp-name***.

  Note that both the OSI Transport Template name and the X25 Access Template name are case sensitive and that NCL is not case sensitive.

- The X.25 counters on your system.

  Use the following command to display the relevant counters on your system:

  ```
  SHOW X25 ACCESS ALL
  ```

  If the counter Outgoing Calls Blocked is incremented each time you make a connect request, it is an outbound X.25 security problem. Refer to Section 17.1.1 for an explanation of the management entities that provide the outbound security.

- The events that your system is issuing.

  Each time an outbound connect request is made to X.25, whether successful or unsuccessful, the X25 Access event Port Terminated is issued. Use the information provided in the event to check the following:

  - Is the Target DTE Address specified in the event correct?

If it is not correct, then the wrong address is being used for the outbound call.

This is because the CONS NSAP to DTE address conversion is not being completed correctly (see Section 17.1.1, step 2).

Check that the information in the X25 Access Reachable Address entity is correct. Check that the CONS NSAP and that the DTE address are correct.

- Is there a Call Association in the event?

  If there is no Call Association, then the system failed to find an X25 Access Template entity, either because the Template does not exist, or the system used the wrong Template (see Section 17.1.1 step 4).

  If there is a Call Association, check that it contains the name of the correct X25 Access Template entity.

- Is the Protocol Identifier correct in the event?

  The Protocol Identifier is specified as the Call Data Value attribute of the X25 Access Template entity. However, the absence of this attribute does not prevent a successful outbound call.

- Is there a DTE Class in the event?

  The DTE Class is specified in the X25 Access Template entity, or X25 Reachable Address entity, see step 5 in Section 17.1.1.

17.3.2 Failed Inbound Connections

An inbound connection can be one of:

- An inbound connection request to the PSDN from an X.25 Gateway Client.
- An inbound connection request from the PSDN to an X.25 Gateway Client.
- An inbound connection request to an X.25 Server from the PSDN.

Start by checking whether the connect request reaches the target system by tracing the connection as described in the Common Trace Facility documentation.

If the trace confirms that the connect request is not being received by the target system, investigate the following:

- That you are using the correct address and where appropriate, subaddress.
- That the X.25 Server for the target system is allowing the X.25 connect request to be received.
- The X.25 counters on the target system.

Use the following command to show the relevant counters on the target machine:

```
NCL> SHOW X25 ACCESS ALL
```

If the counter Incoming Calls Blocked is incremented each time an inbound call is attempted, X.25 security is preventing the call from being received. Check the ACL attribute of the X25 Access Security Filter (see Section 17.1.4 step 3).

If the counter Incoming Calls Failed is incremented each time an inbound call is attempted, there is no X25 Access Filter that matches the incoming connection.

- Check the events that the system is issuing.

  Each time a connect request successfully reaches the system, the X25 Access event Port Terminated is issued. Use the information provided in the event to check the following:

  - Is the Target DTE Address specified in the event the correct address, including the correct subaddress in the case of an X.25 Gateway Client.

    If it is not correct, then the wrong address is being used for the inbound call to the client.

    The fact that the event is received indicates that the correct DTE address is being used.

  - Is there a Call Association in the event?

    If there is no Call Association, then the system failed to find an X25 Access Filter entity because the filter does not exist.

    If the Call Association contains the wrong filter name, X.25 has chosen the wrong filter. For the MTA the filter name should be OSI Transport, and for an X.25 Gateway Client the name of the filter for the X.25 Gateway client.

    Check that the filter being used by X.25 is the one you expected.

    If there is a Call Association, check that it contains the name of the correct X25 Access Filter entity; see Section 17.1.6 step 3.

  - Is the Protocol Identifier correct in the event?

    It should match the Call Data Value attribute of the X25 Access Filter entity.

  - Does the Calling Address Extension match one of the values in the CONS NSAP Address attribute of the OSI Transport entity; see Section 17.1.4 step 4.

18 Problems with Messages

This chapter describes problems that can occur with messages and how you can solve these problems. A message in this context can be an interpersonal message (IPM), a probe, or a report.

Section 18.1 explains the problems that can cause a message to fail. Section 18.2 describes non-delivery reports. Section 18.4 and Section 18.5 describe how to trace the path of a message through your routing domain. Section 18.6 describes how you can trace a probe or report. Section 18.8 describes how an MTA can take responsibility for the messages in a peer MTA's workspace.

18.1 How to Diagnose a Message Failure

If an MTA is unable to transfer or deliver a message, it generates a non-delivery report that explains why the message failed to reach its intended recipient. However, there are several causes of message failure that might not be apparent from a non-delivery report, for example:

- Problems can occur with inconsistencies or omissions in O/R addresses on messages.

  Check the recipient addresses that were used on the message for any obvious omissions. For example, the Country code is mandatory for all O/R addresses.

- Problems can occur with incorrect O/R address information held in the directory.

  Check the directory entries for the recipients on the message. The routing information could be incorrect or the recipients' O/R address entries may have been set up incorrectly.

- A message, probe, or report appears to be lost.

  You may be able to trace its path in your routing domain and find out what has happened to it, see Section 18.4, Section 18.5, and Section 18.6.

Problems with messages and reports are also described by events. All events relating to messages and reports contain the identifier of the message that the event relates to. See Section 18.7 for a description of the events relating to messages and reports.

18.2 Non-delivery Reports

MHS users are most likely to become aware of a problem with a message when they receive a non-delivery report. Table 18–1 lists all the non-delivery reason and diagnostic codes that are listed in the 1992 MHS Standards. Note that the MAILbus 400 MTA does not generate all the non-delivery reason and diagnostic codes listed in Table 18–1

Table 18–1 Non-Delivery Reason and Diagnostic Codes

Code	Diagnostic
Reason Codes	
0	transfer-failure
1	unable-to-transfer
2	conversion-not-performed
3	physical-rendition-not-performed
4	physical-delivery-not-performed
5	restricted-delivery
6	directory-operation-unsuccessful
7	deferred-delivery-not-performed
Diagnostic Codes	
0	unrecognised-OR-name
1	ambiguous-OR-name
2	mts-congestion
3	loop-detected
4	recipient-unavailable
5	maximum-time-expired
6	encoded-information-types-unsupported
7	content-too-long
8	conversion-impractical
9	implicit-conversion-prohibited
10	conversion-not-subscribed
11	invalid-arguments
12	content-syntax-error

(continued on next page)

Table 18–1 (Cont.) Non-Delivery Reason and Diagnostic Codes

Code	Diagnostic
Diagnostic Codes	
13	size-constraint-violation
14	protocol-violation
15	content-type-not-supported
16	too-many-recipients
17	no-bilateral-agreement
18	unsupported-critical-function
19	conversion-with-loss-prohibited
20	line-too-long
21	page-split
22	pictorial-symbol-loss
23	punctuation-symbols-loss
24	alphabetic-symbols-loss
25	multiple-information-loss
26	recipient-reassignment-prohibited
27	redirection-loop-detected
28	dl-expansion-prohibited
29	no-dl-submit-permission
30	dl-expansion-failure
31	physical-rendition-attributes-not-supported
32	undeliverable-mail-physical-delivery-address-incorrect
33	undeliverable-mail-physical-delivery-office-incorrect-or-invalid
34	undeliverable-mail-physical-delivery-address-incomplete
35	undeliverable-mail-recipient-unknown
36	undeliverable-mail-recipient-deceased
37	undeliverable-mail-organization-expired
38	undeliverable-mail-recipient-refused-to-accept
39	undeliverable-mail-recipient-did-not-claim
40	undeliverable-mail-recipient-changed-address-permanently
41	undeliverable-mail-recipient-changed-address-temporarily
42	undeliverable-mail-recipient-changed-temporary-address
43	undeliverable-mail-new-address-unknown
44	undeliverable-mail-recipient-did-not-want-forwarding
45	undeliverable-mail-originator-prohibited-forwarding
46	secure-message-error
47	unable-to-downgrade
48	unable-to-complete-transfer
49	transfer-attempts-limit-reached

The MAILbus 400 MTA generates a non-delivery report for the following reasons:

- The MTA is unable to transfer a message or probe.
- The MTA cannot deliver a message because it is unable to convert an IPM bodypart to a format acceptable to the recipient.

Each non-delivery report contains a reason code, which indicates the reason why the message failed. Non-delivery reports generated by a MAILbus 400 MTA contain one of the following reason codes:

- Unable to Transfer
- Conversion not Performed

Each non-delivery report also contains a diagnostic code that provides additional information about why the message or probe was not transferred or delivered. The reason and diagnostic codes in non-delivery reports that a MAILbus 400 MTA generates conform to the 1992 MHS Standards.

Section 18.2.1 explains the diagnostic codes relating to the Unable to Transfer reason code. Section 18.2.2 explains the diagnostic codes relating to the Conversion not Performed reason code.

18.2.1 Diagnostic Codes Relating to Unable to Transfer

In a non-delivery report generated by a MAILbus 400 MTA, the diagnostic codes relating to the Unable to Transfer reason code are:

- Unrecognised ORname

  This error can be due to one of the following:

  - The routing instruction in the recipient's O/R address entry in the directory explicitly states NONDELIVER.
  - The MTA is unable to find a routing instruction in the directory relevant to the recipient's O/R address.
  - The routing information in the directory has been set up incorrectly. In this case, the MTA also generates the Directory Configuration Error event, which identifies the entries in the directory that are incorrectly set up.
  - The MPDU has been routed to an Agent that is not registered with the MTA. In this case, the MTA also generates the Unknown Agent event.
- Ambiguous ORname

This error can be due to one of the following:

  – The recipient's personal name on the message does not match a unique common name in a complete O/R address entry in the directory (see Section 6.1.2.1). In this case, the MTA also generates the Directory Configuration Error event.
  – The requested delivery method for the recipient does not match the delivery method specified by the recipient's O/R address on the message; for example, the requested delivery method on the message envelope is "teletex-delivery" but the recipient's O/R address is not of the terminal O/R address form.

- MTS Congestion

  This error is due to one of the following:

  – An MPDU has been manually deleted. In this case, the MTA also generates the MPDU Deleted event.
  – A deferred message has been manually deleted. In this case, the MTA also generates the Deferred Message Deleted event.

- Loop Detected

  The MPDU is not being routed efficiently and has either been transferred to the same MTA twice or has been transferred to more than 50 MTAs. When an MTA detects that an MPDU is taking a circular or prolonged route it generates the Loop Detected event.

- Recipient Unavailable

  The recipient of the MPDU is in a different X.400 management domain from the originator and the originator's O/R address entry in the directory has the May Cross CCITT Boundaries attribute set to "False". This attribute value prevents the originator sending messages to recipients in other X.400 management domains.

- Maximum Time Expired

  The MPDU has exceeded its expiry time. The expiry time for MPDUs is specified by the MTA entity in its Local MPDU Expiry Interval attribute and the appropriate priority-based MPDU Expiry Interval attribute. When an MPDU expires within an MTA, the MTA generates the MPDU Expired event.

- Content Too Long

  The content of the MPDU exceeds the acceptable length specified in the Content Information attribute of the recipient's O/R address entry in the directory.

- Invalid Arguments

  This error can be due to one of the following:

  - The MTA cannot determine the form of the recipient's O/R address.
  - The MTA has made three attempts to process the MPDU, all of which failed.
  - The MTA failed to encode the MPDU (this is also an MTA internal inconsistency). In this case, the MTA also generates the Internal Error event.
- Content Syntax Error

  The MTA failed to decode the MPDU's content. The MTA also generates the Invalid MPDU Detected event.
- Protocol Violation

  The MTA failed to decode the MPDU's envelope. The MTA also generates the Invalid MPDU Detected event.
- Too Many Recipients

  The number of recipients for the MPDU exceeds 32,767.
- Unsupported Critical Function

  The MPDU contained a P1 extension whose criticality setting was such that the MTA was obliged to non-deliver the message, according to the P1 protocol. See Appendix B for the MTA's support of P1 extension fields.
- Recipient Reassignment Prohibited

  The routing instruction in the recipient's O/R address entry in the directory specifies RECIPIENT REDIRECT, but the MTA is unable to redirect the message as the recipient-reassignment-prohibited flag is set to "True".
- Redirection Loop Detected

  The recipient already appears in the redirection history field of the MPDU.
- DL Expansion Failure

  Following the expansion of a distribution list, the MPDU envelope contains over 32,767 recipients.
- Unable to Downgrade

  The MTA failed to downgrade the MPDU's envelope.
- Transfer Attempts Limit Reached

The MTA has made three attempts to transfer the MPDU, and has failed each time.

18.2.2 Diagnostic Codes Relating to Conversion Not Performed

Conversion failures are only reported by the MTA that attempts to deliver the message. Attempts by an MTA to convert an IPM bodypart fail because the recipient's requirements regarding IPM bodypart and content types cannot be met.

In a non-delivery report generated by a MAILbus 400 MTA, the diagnostic codes relating to the Conversion not Performed reason code are:

- Encoded Information Types Unsupported

  One or more encoded information types (EITs) in the MPDU are not acceptable to the recipient. The EITs that are acceptable to the recipient are specified in the Content Information attribute of the recipient's O/R address entry in the directory.

- Conversion Impractical

  The MTA tried to convert the IPM bodyparts in the MPDU but the conversion failed. In this case, the MTA also generates the Converter Unavailable event.

- Implicit Conversion Prohibited

  The content information in the recipient's O/R address entry in the directory specifies EITs different from those in the message. However, the MTA was unable to convert the IPM bodyparts in the IPMS content of the MPDU because the implicit-conversion-prohibited flag was set to "True".

- Content Type not Supported

  The content type of the MPDU is not acceptable to the recipient. The content types that are acceptable to the recipient are specified in the Content Information attribute of the recipient's O/R address entry in the directory.

- Conversion with Loss Prohibited

  The MTA was unable to convert the IPM bodyparts in the IPMS content of the MPDU as the conversion could result in data being lost. However, the MTA was prevented from converting the bodypart because the conversion-with-loss-prohibited flag was set to "True".

  During the conversion of a bodypart, data can be lost if there is no direct mapping between the data in the bodypart and the bodypart format acceptable to the recipient, for example, not all Teletex characters can be represented in an IA5 bodypart.

- Unable to Downgrade

  The MTA could not downgrade the IPMS content of the MPDU.

18.2.3 Failure to Receive a Non-Delivery Report

A non-delivery report is delivered to the originator of the message only if the originator requested a report or is able to receive reports.

A user who has requested a non-delivery report may not receive one if:

- The recipient's User Agent cannot receive non-delivery reports or has been set up to not receive non-delivery reports.
- The MTA that detected the problem is unable to generate a non-delivery report.
- An MTA transferring or delivering a non-delivery report to the originator of the message that the report refers to, discards the non-delivery report.

When an MTA is unable to process an incorrectly encoded message, it issues the Invalid MPDU Detected event, places the message in the bad messages directory (see Section 18.3) and attempts to generate a non-delivery report. There are some instances when an MTA either cannot create a non-delivery report or cannot route it to the message originator, for example:

- If the MTA is unable to create a non-delivery report.

  In this case, the MTA issues the Report Generation Failed event.

- If the MTA has created a non-delivery report for the originator, but cannot find a routing instruction in the directory to route the report to the originator.

  In this case, the MTA discards the report it has created and issues the Report Discarded event.

- If the MPDU that the MTA has received is a non-delivery report, and the MTA is unable to decode the MPDU or route it.

  In this case, the MTA copies the MPDU to its bad messages directory and issues an event. If the MTA is unable to decode the MPDU, it issues the Invalid MPDU Detected event. If the MTA is unable to route the MPDU, it issues the Report Discarded event.

18.3 Bad Messages

The MTA copies messages that it cannot process, and reports that it discards, to individual files in its bad messages directory. The name of the file containing the copy of the bad message or report is provided by the event describing the problem. Bodyparts that cannot be converted by the MTA are also copied to the bad messages directory (see Section 18.7.8). Use the Message Decoder tool to examine messages, reports, and IPM bodyparts in the MTA's bad messages directory.

For the location of the MTA's bad messages directory and information about how to run the Message Decoder tool, refer to the appendix describing the operating system specific information.

18.4 Tracing a Message

You may need to find out what has happened to a message in order to solve some problems with a message that a user has received or when a user complains that a message has not been delivered.

If an MTA fails to transfer, deliver, or export a message, it provides some information about what has happened to the message through events or a non-delivery report. If, for any reason, you cannot find out what has happened to a message, then trace its path in your routing domain or as far as you can.

Processed Message entities, which are automatically created by an MTA, hold information about the messages that the MTA has received. A Processed Message entity describes what the MTA is doing or has done to a particular message. When the message leaves the MTA, the Processed Message entity identifies the next destination of the message, provided that Message History logging is enabled. Using the information held in a Processed Message entity you can trace messages in your routing domain.

Before you can trace a message you need to:

- Have access to a privileged account.

  For information about the privileges that you need, refer to the appendix describing the operating system specific information.

- Know the identifier of the message that you want to trace.

  This is provided by the originator's User Agent or the Gateway that imported the message.

Note

To trace the path of a message through your routing domain you must have Message History logging enabled at each MTA in your routing domain. If you have not, then you can only trace a message at the MTA that is handling the message or at MTAs that have Message History logging enabled. See Chapter 10 for information on how to use Message History logging.

To help you decide whether or not to trace a message you need to find out the following information:

- Who originated or forwarded the message.
- The name of the node where the message entered your routing domain.

  This is either the node where the originator's MTA is running or where a Gateway or boundary MTA is running.
- The day and time when the message was sent.
- The name of each intended recipient of the message.

Using this information, you need to consider if there has been enough time for the message to have arrived. Allow for any known network problems, such as a node being unavailable, network failure, or busy periods.

If you consider that the message should have arrived, then start to trace its path through your routing domain. It is best to start tracing a message at the first MTA that handled the message in your routing domain. This is because this MTA is able to provide a complete list of all the recipients of the message. Other MTAs that receive the message from the first MTA may only know about some of the recipients.

Use the following command to display information about the message:

```
SHOW NODE "node-id" MTA PROCESSED MESSAGE [COUNTRY = "country",-
ADMINISTRATION DOMAIN = "admd", PRIVATE DOMAIN = "prmd",-
LOCAL IDENTIFIER = "local-id"] ALL ATTRIBUTES
```

where:

- *country* is the Country name.
- *admd* is the name of the Administration Management Domain.
- *prmd* is the name of the Private Management Domain.
- *local-id* is a local identifier.

If there is no record of the message in the MTA, then you can also look for the message in the MTA's deferred messages workspace. Use the following command to display the message in the deferred messages workspace:

```
SHOW NODE "node-id" MTA DEFERRED MESSAGE [COUNTRY = "country",-
ADMINISTRATION DOMAIN = "admd", PRIVATE DOMAIN = "prmd",-
LOCAL IDENTIFIER = "local-id"] ALL ATTRIBUTES
```

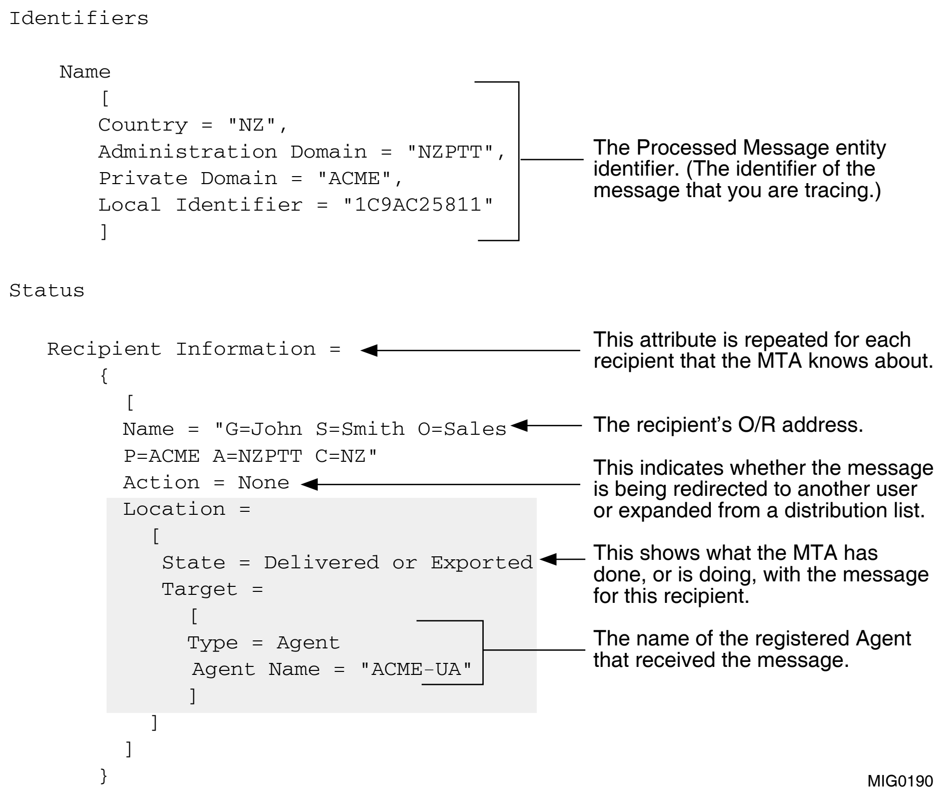
where:

- *country* is the Country name.
- *admd* is the name of the Administration Management Domain.
- *prmd* is the name of the Private Management Domain.
- *local-id* is a local identifier.

See Section 18.5 for more information about deferred messages.

If there is no record of the message in the MTA or in its deferred messages workspace, then you may not be able to trace the message. See Section 18.4.6 for information about how to proceed with the message trace.

If there is a record of the message in the MTA, then the following is an example of a response from the command you issued:



```
Identifiers

    Name
       [
       Country = "NZ",
       Administration Domain = "NZPTT",
       Private Domain = "ACME",
       Local Identifier = "1C9AC25811"
       ]

Status

   Recipient Information =
       {
         [
         Name = "G=John S=Smith O=Sales
         P=ACME A=NZPTT C=NZ"
         Action = None
         Location =
           [
            State = Delivered or Exported
            Target =
              [
              Type = Agent
               Agent Name = "ACME-UA"
              ]
           ]
         ]
       }
```

The Processed Message entity identifier. (The identifier of the message that you are tracing.)
This attribute is repeated for each recipient that the MTA knows about.
The recipient's O/R address.
This indicates whether the message is being redirected to another user or expanded from a distribution list.
This shows what the MTA has done, or is doing, with the message for this recipient.
The name of the registered Agent that received the message.
MIG0190

The information that you need to trace the path of the message is in the Location field (the shaded area in the example). The Location field is subdivided into the following fields:

- State

  This describes what the MTA is doing or has done with the message; for example, Being Processed.

- Target

  This is the next destination of the message, if known by the MTA.

- Type

  This identifies the target, which is one of the following:

  - Agent

    A registered Agent of the MTA. This is identified by the name of the Agent entity.

  - Mailbox

    An unregistered Agent of the MTA. This is an O/R address.

  - Domain

    Another routing domain. This is identified by the name of the Domain entity in the MTS module. This name also corresponds to the value of the Peer Domain attribute of the Peer MTA entity that represents a peer MTA in that routing domain.

  - MTA

    A peer MTA in your routing domain. This is identified by the name of the peer MTA.

If an MTA has yet to process the message and determine its target, then Unknown is displayed in the Type field.

The information in the State and Type fields indicates one of the following:

- The message is within the MTA (Section 18.4.1).
- The message has been manually deleted (Section 18.4.2).
- The message has expired and has not been delivered (Section 18.4.2).
- The message has been corrupted and cannot be processed by the MTA (Section 18.4.2).
- The message has been delivered or exported to an Agent (Section 18.4.3).
- The message has been transferred to a peer MTA in your routing domain (Section 18.4.4).
- The message has been transferred to a peer MTA in another routing domain (Section 18.4.5).

18.4.1 Messages in the MTA

The following combinations of State and Type field values of the Recipient Information attribute indicate that the message is within this MTA or in the process of leaving this MTA:

State = Awaiting Processing	Type = Unknown
State = Being Processed	Type = Unknown
State = Awaiting Processing Retry	Type = Unknown
State = Awaiting Transfer	Type = MTA
State = Awaiting Transfer	Type = Domain
State = Being Transferred	Type = MTA
State = Being Transferred	Type = Domain
State = Awaiting Transfer Retry	Type = MTA
State = Awaiting Transfer Retry	Type = Domain
State = Awaiting Delivery or Export	Type = Agent
State = Awaiting Delivery or Export	Type = Mailbox
State = Being Delivered or Exported	Type = Agent
State = Being Delivered or Exported	Type = Mailbox
State = Awaiting Delivery or Export Retry	Type = Agent
State = Awaiting Delivery or Export Retry	Type = Mailbox

The identity of the target, if known by the MTA, is also shown in the Type field.

Messages that are Awaiting Transfer Retry may be delayed in the MTA because the MTA is unable to set up an association to a peer MTA. Find out if there are any relevant events, such as the Outbound Soft Rejection event, in the event sink from the MTA. Take the appropriate action for the event as described in Chapter 16.

Messages that are Awaiting Delivery or Export Retry may be delayed in the MTA because the MTA is unable to connect to an Agent or Mailbox. Find out if there are any occurrences of the Rejected Agent Connection or Unknown Agent events in the event sink from the MTA. Take the appropriate action for these events as described in Chapter 19.

All other states indicate that the MTA is processing the message normally. You can repeat the trace later on to see if the MTA has finished processing the message.

18.4.2 Deleted, Expired, or Corrupt Messages

When a message has expired, has been corrupted, or has been manually deleted, the State field of the Recipient Information attribute contains one of the following values:

- Deleted

  When a message is manually deleted, the MTA generates the MPDU Deleted event (Section 18.7.7). The MTA also sends a non-delivery report to the originator of the message.

- Expired

  When a message expires, the MTA generates the MPDU Expired event (Section 18.7.3) and sends a non-delivery report to the originator of the message.

- Corrupt

  When the MTA receives a message that has been corrupted, it copies it to a file in the bad messages directory and generates the Invalid MPDU Detected event (see Section 18.7.2).

The Type field of the Recipient Information attribute contains the target as appropriate.

Note that when the MTA generates a non-delivery report, the non-delivery report is delivered to the originator of the message only if the originator requested a report or is able to receive reports.

18.4.3 Messages Sent to an Agent

If the MTA has exported or delivered the message to a registered Agent, the State and Type fields of the Recipient Information attribute contain:

```
State = Delivered or Exported          Type = Agent
```

The name of the Agent is also shown in the Type field.

In this case, the MTA no longer has responsibility for the message. The action you take depends on the type of interface used by the Agent:

- The Agent is a Gateway that uses the Shared File interface

  The message could be awaiting collection by the Agent. The MTA exports a message through the Shared File interface by placing the message into a file and storing the file in the Output queue for collection by the Agent. The Output queue for a particular Agent that uses the Shared File interface is in the MTA's workspace directory. Refer to the appendix describing the operating system specific information for the location of the MTA's workspace.

If messages stay in the Output queue for some time, then check with the manager of the Agent that the Agent is running correctly.

Agents that are unable to process messages that they retrieve from the Output queue may rename the file in the Output queue so that it is prefixed with the letter "U". The letter "U" indicates that the Agent cannot deliver the message. The Agent writes an error message into its log file identifying the file containing the message and the reason it was unable to deliver the message. Contact the person responsible for managing the Gateway or consult the Gateway documentation.

You can examine a file in the Output queue using the Message Decoder tool. See the appropriate appendix for the commands to run the Message Decoder tool.

If the message is not in a file in the Output queue, then make further enquiries at the Gateway. See the relevant Gateway documentation or contact the person responsible for managing the Gateway.

- The Agent uses the XAPI interface.

  Make further enquiries at the Agent. See the relevant Agent documentation or contact the person responsible for managing the Agent.

If the MTA has delivered the message to an unregistered Agent, the State and Type fields of the Recipient Information attribute contain:

```
State = Delivered or Exported          Type = Mailbox
```

The O/R address of the person using the unregistered Agent is also shown in the Type field.

In this case, the MTA no longer has responsibility for the message. Make further enquiries at the unregistered Agent. An unregistered Agent serves only one O/R address, so the User Agent must have received the message.

18.4.4 Messages Transferred to a Peer MTA in the Same Routing Domain

If the MTA has transferred the message to a peer MTA in your routing domain, the State and Type fields of the Recipient Information attribute contain:

```
State = Transferred                    Type = MTA
```

The name of the peer MTA is also shown in the Type field.

In this case, this MTA no longer has responsibility for the message, continue tracing the message at the peer MTA. To continue the trace at the peer MTA, find out the name of the node where the peer MTA is running. Display the Processed Message entity at this peer MTA and find out what the peer MTA is doing or has done with the message.

18.4.5 Messages Transferred to Another Routing Domain

If the MTA has transferred the message to a peer MTA in another routing domain, then the State and Type fields of the Recipient Information attribute contain:

```
State = Transferred          Type = Domain
```

The name of the routing domain where the peer MTA is located is also shown in the Type field.

In this case, this MTA no longer has responsibility for the message.

You cannot trace messages in another routing domain, unless you have management responsibility for that routing domain. However, if the other routing domain contains MAILbus 400 MTAs, it is possible for the person managing that routing domain to continue tracing the message.

18.4.6 Problems Tracing Messages

If you are unsuccessful when you first try to trace a message, continue again at another MTA; for example, the recipient's MTA. If another MTA in your routing domain is able to display information relating to the message, then you can continue the trace.

If you are unable to trace a message in your routing domain it could be that there is no information about that message in your routing domain or the Message History information at a particular MTA is missing or incomplete. The most likely reason for information about a message to be missing from your routing domain is because Message History logging is not enabled or, if it is, the relevant information has been purged. Also, information about the message may not be available for to one of the following reasons:

- Message History logging failed.

  In this case, check the event sinks for occurrences of the Message History Data Lost event. See Section 21.2.2 for information about this event, and take the appropriate action.

- The first MTA that should have handled the message in your routing domain never received the message.

This could be due to one of the following:

- A Gateway that uses the Shared File interface fails to send the message to the MTA.

  The message could be awaiting collection by the MTA. Agents that use the Shared File interface send a message to the MTA by placing a file containing the message in the Input queue. The Input queue for a particular Agent that uses the Shared File interface is in the MTA's workspace directory. Refer to the appendix describing the operating system specific information for the location of the MTA's workspace. If the MTA is unable to retrieve the file containing the message from the Input queue, then the MTA generates the System Interface Error event reporting a file access failure (Section 20.1).

  You can examine a message in the Input queue using the Message Decoder tool. See the appropriate appendix for the commands to run the Message Decoder tool.

  If the message is not in the Input queue, then contact the person responsible for managing the Gateway.

- An Agent that uses the XAPI interface failed to send the message to the MTA.

  Contact the person responsible for managing the Agent.

- A peer MTA in another routing domain failed to send the message to the MTA.

  Contact the person responsible for managing the peer MTA.

• The MTA did not log Message History information about the message because the message was recovered from its workspace by another MTA in the same routing domain (see Section 18.8).

  In this case, the MTA that recovered the workspace copied the message to its own workspace and is able to provide information about the message. Therefore, resume tracing the message at the MTA that copied it. Note that if the workspace was recovered by several MTAs it is not possible to find out which messages were copied by a particular MTA. In this case, you have to attempt to trace the message at each MTA that copied messages from the workspace until you find an MTA that has information about the message.

If, after all your investigations are complete, you are still unable to find out what has happened to the message, then complete the following steps:

1. Make sure Message History logging is enabled at all the MTAs in your routing domain, if you have not already done so.

2. Ask the originator of the lost message to re-send it.

If this message fails to be delivered, then you can trace what has happened to it.

18.5 Tracing Deferred Messages

A deferred message is a message that is processed by the originator's MTA at a specified time after it has been submitted by the originator's User Agent. The originator specifies the time when processing is to begin. When the MTA receives the message, it stores the message in its deferred messages workspace. The MTA retrieves the message for processing at the time specified by the originator. The MTA creates a Deferred Message entity for each message in its deferred messages workspace.

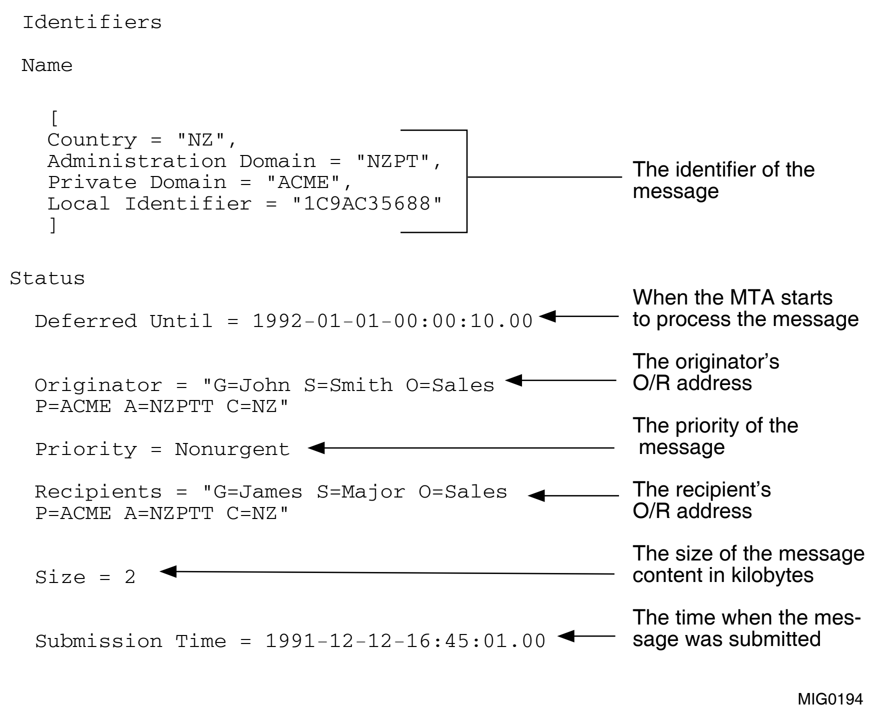
It may be that a message you are trying to trace is in the MTA's deferred messages workspace. Use the following command to display information about a specific deferred message:

```
SHOW NODE "node-id" MTA DEFERRED MESSAGE [COUNTRY = "country",-
ADMINISTRATION DOMAIN = "admd", PRIVATE DOMAIN = "prmd",-
LOCAL IDENTIFIER = "local-id"] ALL ATTRIBUTES
```

where:

- *country* is the Country name.
- *admd* is the name of the Administration Management Domain.
- *prmd* is the name of the Private Management Domain.
- *local-id* is a local identifier.

The following is an example of a response from this command:



```
Identifiers

Name

  [
  Country = "NZ",
  Administration Domain = "NZPT",
  Private Domain = "ACME",
  Local Identifier = "1C9AC35688"
  ]

Status

  Deferred Until = 1992-01-01-00:00:10.00

  Originator = "G=John S=Smith O=Sales
  P=ACME A=NZPTT C=NZ"

  Priority = Nonurgent

  Recipients = "G=James S=Major O=Sales
  P=ACME A=NZPTT C=NZ"

  Size = 2

  Submission Time = 1991-12-12-16:45:01.00
```

The identifier of the message
When the MTA starts to process the message
The originator's O/R address
The priority of the message
The recipient's O/R address
The size of the message content in kilobytes
The time when the message was submitted
MIG0194

18.6 Tracing Probes and Reports

You can trace probes and reports while they are being processed by the MTA. You do this by using the MPDU entity. The Type attribute in the MPDU entity indicates whether the MPDU is related to a message, probe, or report.

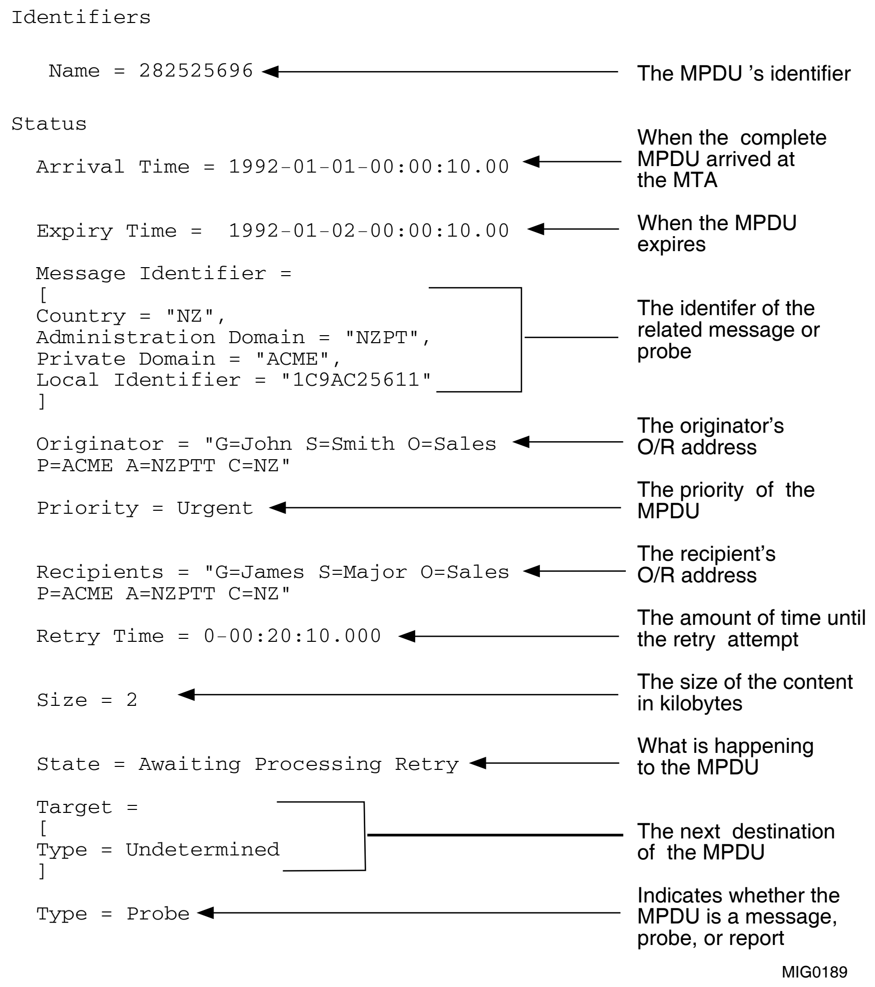
Use the following command to display all the MPDUs in the MTA that relate to reports or probes:

```
SHOW NODE "node-id" MTA MPDU *, WITH TYPE = value
```

where *value* is either `PROBE` or `REPORT`.

If there are any probes or reports in the MTA, this command displays the related MPDU entity for each probe or report.

The following is an example of a typical MPDU entity:



```
Identifiers

   Name = 282525696

Status

  Arrival Time = 1992-01-01-00:00:10.00

  Expiry Time =  1992-01-02-00:00:10.00

  Message Identifier =
  [
  Country = "NZ",
  Administration Domain = "NZPT",
  Private Domain = "ACME",
  Local Identifier = "1C9AC25611"
  ]

  Originator = "G=John S=Smith O=Sales
  P=ACME A=NZPTT C=NZ"

  Priority = Urgent

  Recipients = "G=James S=Major O=Sales
  P=ACME A=NZPTT C=NZ"

  Retry Time = 0-00:20:10.000

  Size = 2

  State = Awaiting Processing Retry

  Target =
  [
  Type = Undetermined
  ]

  Type = Probe
```

The MPDU 's identifier
When the complete MPDU arrived at the MTA
When the MPDU expires
The identifer of the related message or probe
The originator's O/R address
The priority of the MPDU
The recipient's O/R address
The amount of time until the retry attempt
The size of the content in kilobytes
What is happening to the MPDU
The next destination of the MPDU
Indicates whether the MPDU is a message, probe, or report
MIG0189

18.7 Events Related to Problems with Messages

The following events are related to problems with messages or MPDUs:

- Expiry Alarm Threshold Exceeded (Section 18.7.1)
- Invalid MPDU Detected (Section 18.7.2)
- MPDU Expired (Section 18.7.3)
- Report Discarded (Section 18.7.4)
- Report Generation Failed (Section 18.7.5)
- Deferred Message Deleted (Section 18.7.6)
- MPDU Deleted (Section 18.7.7)
- Converter Unavailable (Section 18.7.8)
- Recovery Finished (Section 18.8.4)

18.7.1 Expiry Alarm Threshold Exceeded

This event is a warning that the MPDU could expire and not be delivered, and occurs whenever an MPDU has been in the MTA for half the time specified in the Local MPDU Expiry Interval attribute. This event is counted by the MTA entity's Expiry Alarms counter.

The event provides the following information:

```
Name = mpdu-id
State = mpdu-state
Target = target
```

where:

- *mpdu-id* is the identifier of the MPDU that is likely to expire.
- *mpdu-state* is the location of the MPDU in the MTA.
- *target* is the next destination of the MPDU.

Action

Action is not necessary in isolated instances of this event.

If you receive several of these events each identifying the same target, then there could be problem with that particular target. See Section 18.4 for a description of how to identify a target. You can check the event sink for other events related to the same target; for example, if a peer MTA is unavailable, then there might be events relating to the failure or rejection of an association to that particular peer MTA.

If you receive several of these events that each identify different targets, then there could be a problem with the MTA itself; for example the MTA has reached its limit of outbound associations and cannot set up new associations. Check the event sink for events that indicate congestion. See Section 7.2.3 for a list of these events.

If this event occurs frequently because the MTA is congested, then tune the MTA to improve the throughput of messages, see Section 7.3.

18.7.2 Invalid MPDU Detected

This event occurs whenever an MTA or Gateway cannot process an MPDU because the MPDU is incorrectly encoded. The MTA copies the MPDU to its bad messages directory. This event is counted by the MTA entity's Invalid MPDUs Detected counter.

The event provides the following information:

```
Name = mpdu-id
Message Identifier = message-id
Source = source
Saved MPDU = file-spec
Reason = reason
Diagnostic = diagnostic
Supplementary Information = supplement
```

where:

- *mpdu-id* is the locally assigned identifier of the invalid MPDU.
- *message-id* is the identifier of the related message, probe, or report.
- *source* is the name of the Domain, peer MTA, or Agent where the MPDU originated. If the Source is a Mailbox, this field contains an O/R address, or is null if the O/R address is incorrectly encoded.
- *file-spec* is the name of the file in the bad messages directory that contains a copy of the MPDU.
- *reason* is the reason why the MPDU was invalid. This information is the appropriate non-delivery text specified in the 1992 MHS Standards. Reasons are listed in Section 18.2.1.
- *diagnostic* is information that identifies the error that caused the MPDU to be rejected as invalid. This information is the appropriate diagnostic text specified in the 1992 MHS Standards. Diagnostics are listed in Section 18.2.1.

- *supplement* is additional information about the problem added by an MTA. Note that this field can be empty.

Action

Investigate every occurrence of this event, which could be due to a file or device error, or to a protocol error in the MPDU.

Examine the saved copy of the MPDU to identify the problem. Use the Message Decoder tool to display the saved copy of the MPDU in the MTA's bad messages directory. For information about how to run the Message Decoder tool, refer to the appendix describing the operating system specific information.

If possible, inform the originator that the message, probe, or report failed.

In the case of the Supplementary Information code 140 (GDI in message identifier does not match the GDI in the first trace element), check the Global Domain Identifiers attribute of the MTS entity. If your routing domain is multihomed, make sure that all the synonyms for the GDI are included in the Global Domain Identifiers attribute of the MTS entity, an entity of the MTS module.

18.7.3 MPDU Expired

This event occurs whenever an MPDU expires. When an MPDU expires, the MTA sends a non-delivery report to the originator of the message. Note that the non-delivery report is delivered to the originator of the message only if the originator requested a report or is able to receive reports. This event is counted by the MTA entity's Expired MPDUs counter.

The event provides the following information:

```
Name = mpdu-id
Message Identifier = message-id
State = mpdu-state
```

where:

- *mpdu-id* is the locally assigned identifier of the MPDU.
- *message-id* is the identifier of the related message, probe, or report.
- *mpdu-state* is the location of the MPDU in the MTA at the time it expired.

Action

Investigate every occurrence of this event, because it could indicate that other problems are occurring in the MTA or in your routing domain which are causing delays in the flow of messages. If this event occurs frequently, then you may need to tune the MTA to improve the flow of messages through it, see Section 7.3.

18.7.4 Report Discarded

This event occurs whenever an MTA discards a report because it is unable to route it correctly. If the MTA generated the report, then the MTA copies the the report to the bad messages directory. If the report was transferred in from a peer MTA, then the MTA copies the report to the bad messages directory. This event is counted by the MTA entity's Reports Discarded counter.

The event provides the following information:

```
Name = mpdu-id
Original Message Identifier = message-id
Reason = reason
Diagnostic = diagnostic
Supplementary Information = supplement
Saved MPDU = file-spec
```

where:

- `mpdu-id` is the locally assigned identifier of the MPDU that contained the report.
- `message-id` is the identifier of the original message or probe that the report was referring to.
- `reason` is the reason why the delivery of the subject of the report failed. This information is the appropriate non-delivery text specified in the 1992 MHS Standards. Reasons are listed in Section 18.2.1.
- `diagnostic` is information that identifies the error that caused the report to be discarded. This information is the appropriate non-delivery diagnostic text specified in the 1992 MHS Standards. Diagnostics are listed in Section 18.2.1.
- `supplement` is additional information about the problem added by an MTA. Note that this field can be empty.
- `file-spec` is the name of the file in the MTA's bad messages directory, that contains a copy of the message or probe that the discarded report referred to, or, in the case of received reports, the report itself.

Action

Use the Message Decoder tool to display the saved copy in the MTA's bad messages directory. For information about how to run the Message Decoder tool, refer to the appendix describing the operating system specific information. Examine the saved copy in the MTA's bad messages directory. If the copy is a message or probe, then identify the originator. If possible, notify the originator that their message or probe failed. If the copy in the bad messages directory is a report, then notify the intended recipient of the report about the failure of the message or probe that the report refers to.

18.7.5 Report Generation Failed

This event occurs whenever an MTA is unable to create a report; for example, a non-delivery report. This event is counted by the MTA entity's Report Generation Failures counter.

The event provides the following information:

```
Name = mpdu-id
Original Message Identifier = message-id
```

where:

- *mpdu-id* is the locally assigned identifier of the MPDU.
- *message-id* is the identifier of the original message or probe for which a report could not be generated. Note that the identifier of the original message is only available if it was correctly encoded.

Action

This event is always preceded by the Invalid MPDU Detected event referring to the same message or probe. Check the event sink for an Invalid MPDU Detected event relating to the same MPDU; see Section 18.7.2 for information about this event.

If possible, inform the originator that the message or probe failed.

18.7.6 Deferred Message Deleted

This event occurs whenever a message is manually deleted from an MTA's deferred messages workspace. This event is counted by the MTA entity's Deleted Deferred Messages counter.

The event contains the identifier of the message that was deleted.

Action

It is not normally necessary to investigate this event. The MTA automatically sends a non-delivery report to the originator of the message. Note that the non-delivery report is delivered to the originator of the message only if the originator requested a report or is able to receive reports.

18.7.7 MPDU Deleted

This event occurs whenever an MPDU is manually deleted. This event is counted by the MTA entity's Deleted MPDUs counter.

The event provides the following information:

```
Name = mpdu-id
Message Identifier = message-id
State = mpdu-state
```

where:

- *mpdu-id* is the locally assigned identifier of the deleted MPDU.
- *message-id* is the identifier of the message, probe, or report that has been deleted.
- *mpdu-state* is the location of the MPDU in the MTA at the time it was deleted.

Action

It is not normally necessary to investigate this event. If the MPDU is a message or a probe, then the MTA automatically sends a non-delivery report to the originator. Note that the non-delivery report is delivered to the originator of the message only if the originator requested a report or is able to receive reports.

18.7.8 Converter Unavailable

This event is generated when a converter fails to convert a bodypart or when the MTA is unable to locate the converter image it needs.

Failure by an MTA to convert a bodypart does not necessarily mean that the message is not delivered. For example, an MTA can transfer a message to a peer MTA without converting an unacceptable bodypart. However, if the MTA is to deliver or export the message to an Agent, then delivery or export depends on whether the Agent can receive any bodypart. If the Agent can receive any bodypart, then the MTA delivers or exports the message. Otherwise, the MTA issues a non-delivery report. See Section 3.8.1 for a description of how an MTA decides whether or not to convert a bodypart.

This event is counted by the MTA entity's Unavailable Converters counter.

The event provides the following information about the problem:

```
Converter Name = name
Step = step
Error Log = log
Saved Bodypart = file-spec
Message Identifier = message-id
Source = source
Originator = oraddress
```

where:

- `name` is the name of the Converter entity that describes the converter or a sequence of conversions.
- `step` is the name of a Converter entity that describes the converter image that failed during a sequence of conversions. Note that this field only has a value when the Converter entity in the Converter Name field describes a sequence of conversions.
- `log` is the file specification of the error log containing text error messages from the converter that failed.

  Note that the error log is only available if the converter fails and sends a fatal error message to the MTA.

  Some types of error reported in the log indicate a serious problem with the MTA. These error messages are labeled `Internal error`. Report these errors to HP. See Chapter 23 for information about how to contact HP and the information that you need to supply.
- `file-spec` is the name of the file containing a copy of the bodypart. This copy is in ASN.1 (BER) as described in Section 12.2.2.

  Use the Message Decoder tool to examine the saved copy of the bodypart that was being converted. Refer to the appendix describing the operating system specific information for information about running this tool.
- `message-id` is the identifier of the message containing the bodypart that the MTA sent to the converter.
- `source` is the identity of the Agent, peer MTA, or domain that sent the message to the MTA.
- `oraddress` is the O/R address of the originator of the message.

This is followed by one of:

1 `Error = Conversion Failed`

2 `Error = Conversion Result Does Not Match Target`

3 `Error = Cannot Decode Resultant Bodypart`

4 `Error = Converter Image Does Not Exist or is Inaccessible`

Some of the problems identified by this event are caused by Converter and Bodypart entities being incorrectly set up. Entities that describe the converters supplied with the MTA, and the bodyparts referenced by those converters, are created by commands in the MTA's startup script.

If there is a problem with a HP-defined converter and you suspect the problem was caused because you have amended the MTA's startup script, then return to using the original version of the script.

If you are unable to find out why a HP-defined converter has failed or why a particular bodypart cannot be converted, then contact HP.

Action

1. The converter failed to recognize the source bodypart or failed before it produced a target bodypart.

   This is due to one of the following:

   - The converter received a bodypart that was incorrectly encoded.

     Problems with incorrect encoding of the source bodypart are due to the original encoding by the User Agent being incorrect. You need to identify the originator's User Agent and check that it encodes the characters and structure of the bodypart correctly.

   - The Source attribute in the Converter entity that describes the converter has been incorrectly set up.

     The MTA sends the converter the bodypart type specified by the Converter entity's Source attribute. If this attribute has been incorrectly set up, then the MTA sends the wrong bodypart type to the converter.

     If the converter is a HP-defined converter and you are using the MTA's startup script to create the Converter entity, make sure that you are using the startup script that was supplied with the MTA. If the converter is one that you have defined, then check that the Source attribute of the Converter entity has been correctly set up.

     Use the following command to display all the Converter entity's attributes:

     ```
     SHOW NODE "node-id" MTA CONVERTER "name" ALL ATTRIBUTES
     ```

     where *name* is the name of the Converter entity.

If the Source attribute describes a bodypart that you have defined, then check the Bodypart entity that describes the bodypart. Use the following command to display the Bodypart entity:

```
SHOW NODE "node-id" MTA BODYPART "name" ALL ATTRIBUTES
```

where *name* is the name of the Bodypart entity specified in the Source attribute of the Converter entity.

Check the Source bodypart definition against the specification of the converter. Also, check that the converter is using the correct object identifiers for the data and parameter components of the bodypart.

If the Source attribute is incorrect, delete the Converter entity and create a new entity containing the correct value.

If the Source bodypart definition is incorrect, delete the Bodypart entity and create a new entity containing the correct attributes values. Add any corrections you make to your Bodypart and Converter entities to the MTA's startup script.

- There is a decoding error in the converter.

  If you cannot find a problem in the way the Source attribute is set up, the problem might be with the converter image. You need to get the converter checked by the supplier of the converter image. Note there is a copy of the bodypart in the file specified by the event.

2 The converter has incorrectly converted the source bodypart.

The converter has generated a bodypart that the MTA is not expecting to receive from this converter. This is caused by one of the following:

- The Target attribute of the Converter entity that describes the converter has been incorrectly set up.

  Display all the attributes of the Converter entity using the following command:

  ```
  SHOW NODE "node-id" MTA CONVERTER "name" ALL ATTRIBUTES
  ```

  where *name* is the name of the Converter entity.

  If the Target attribute describes a bodypart that you have defined, then check the Bodypart entity that describes your bodypart. Use the following command to display the Bodypart entity:

  ```
  SHOW NODE "node-id" MTA BODYPART "name" ALL ATTRIBUTES
  ```

  where *name* is the name of the Bodypart entity specified in the Target attribute of the Converter entity.

Check the Target bodypart definition against the specification of the converter. Check that the converter is using the correct object identifiers for the data and parameter components of the bodypart.

If the Target attribute is incorrect, delete the Converter entity and create a new entity containing the correct value.

If the Target bodypart definition is incorrect, delete the Bodypart entity and create a new entity containing the correct attributes values. Add any corrections you make to your Bodypart and Converter entities to the MTA's startup script.

- There is an encoding error in the converter itself.

  If you cannot find a problem in the way the Target attribute is set up, the problem might be with the converter image. You need to get the converter checked by the supplier of the converter image. Note there is a copy of the bodypart in the file specified by the event.

3 The MTA is unable to decode the bodypart that has been generated by the converter.

This is due to an error in the encoding of the Target bodypart by the converter image. The Target bodypart could have been incorrectly encoded for one of the following reasons:

- The converter received a bodypart that was incorrectly encoded.
- There is an encoding error in the converter itself.
- There is a problem with the MTA.

If this problem occurs occasionally, then it is probably due to incorrect encoding of the source bodypart by the User Agent that submitted the message. You need to identify the originator's User Agent and check that it encodes the characters and structure of the bodypart correctly.

If this error occurs frequently then there could be an encoding error in the converter. If the converter is one that you have defined, then check the converter encoding.

If you cannot find an error in the encoding of a converter that you have defined, then contact HP. If the converter is a HP-defined converter, then contact HP. See Chapter 23 for information about how to contact HP and the information that you need to supply.

4 The converter image is not accessible by the MTA.

Check whether the converter image file is correctly located in the MTA's converter image directory. For the location of this directory, refer to the appendix describing the operating system specific information. If the converter image is in the MTA's converter image directory, then check that the file protection and ownership are correct. Also check that the name of the Converter entity exactly matches the name of the converter image that it describes.

OpenVMS

On OpenVMS systems, the name of a converter image has a file extension of .EXE. Do not include the .EXE extension in the name of the corresponding Converter entity.

♦

If you are unable to find a HP-defined converter image, then reinstall the MTA.

If you are unable to find a converter image that you have defined, then reinstall your converter in the MTA's converter image directory.

18.8 Recovering Messages From an MTA's Workspace

There might be occasions when you need to remove an MTA from the network, for example, because the node where the MTA is running is shut down for maintenance. When an MTA is shut down, messages remain in its workspace and cannot be processed until the MTA is recreated. However, another MTA can access the MTA's workspace and take responsibility for the messages by copying them to its own workspace. When an MTA copies a message from a peer MTA's workspace it attempts to process the message as the peer MTA would have done. The copying of messages by an MTA from a peer MTA's workspace is called recovery.

Before the MTA starts to copy a message from a peer MTA's workspace, it **locks** the peer MTA's workspace. Locking controls access to the peer MTA's workspace, so that only one MTA can access a particular message at any one time. As access to the peer MTA's workspace is controlled, more than one MTA can recover the peer MTA's workspace and the peer MTA can be recreated at any time during the recovery process.

If an MTA cannot lock the peer MTA's workspace, recovery can still take place but access to the messages in the workspace is not controlled. In this case, when you issue the Recover command, it returns a warning text informing you that the peer MTA's workspace is not locked (see Section 18.8.3).

Messages copied from the peer MTA's workspace that can only be delivered, exported, or transferred by that peer MTA, cannot reach their intended recipients. This is because the MTA that copied the messages does not have the correct Agent or Peer MTA entities. If an MTA copies a message from a peer MTA's workspace that can only be delivered, exported, or transferred by the peer MTA, then, if the peer MTA remains unavailable, the message eventually expires. When the message expires, the MTA attempts to send a non-delivery report to the originator of the message.

18.8.1 Limitations to Using Recovery

Recovery of a peer MTA's workspace depends on the following conditions:

- The Recover command can be issued only from a privileged account. See the appropriate appendix for information about privileges.
- Only MTAs in the same routing domain can recover messages from each other's workspace.
- An MTA can only recover messages from one peer MTA's workspace at a time.
- Before an MTA can recover messages from a peer MTA's workspace it must have read, write, and delete access to the workspace.
- An MTA running on Tru64 UNIX cannot access a workspace on OpenVMS.
- An MTA running on OpenVMS cannot access a workspace on Tru64 UNIX.

If the MTA is unable to lock the peer MTA's workspace, the following additional restrictions also apply:

- Only one MTA is able to recover messages from the peer MTA's workspace.
- Do not recreate the peer MTA whose messages are being recovered until recovery finishes.

18.8.2 Setting Up Recovery of an MTA's Workspace

You need to find out the file specification of the workspace you want to recover and then set up read, write, and delete access to that workspace. See the appropriate appendix for the location of an MTA's workspace.

The following are examples of how you can set up access to the peer MTA's workspace:

Tru64 UNIX

On Tru64 UNIX systems, access has to be set up at the MTA that will recover the messages and at the peer MTA whose messages you want to recover.

If you want to use the Network File System (NFS) to set up access, then:

- NFS mount the peer MTA's workspace by root at the MTA doing the recovery.
- Export the peer MTA's workspace using the /etc/exports file to allow client superuser access.

♦

OpenVMS

On OpenVMS systems, if the peer MTA is located on a node that is part of a cluster, and each MTA in the cluster has access to the same disks, you can issue the Recover command at any node in the cluster where an MTA is running.

If all the nodes where MTAs are running share a common SYSUAF.DAT file, no additional security set up is required.

If the peer MTA whose messages are to be recovered is not on a node in a cluster, you need access to the peer MTA's SYSTEM account at each MTA where you issue the Recover command. Alternatively, you can set up a network proxy on the peer MTA's node so that an MTA can access the peer MTA's workspace.
♦

If it is not possible for an MTA to access the peer MTA's workspace over the network, you might consider moving the device containing the peer MTA's workspace so that it is local to the MTA that is to recover the messages. Alternatively, you can backup the peer MTA's workspace and copy it to the MTA that is to recover the peer MTA's messages. Note that in this case, never recreate the peer MTA whose messages have been recovered. If you do recreate the peer MTA, duplicate messages could enter the MTS.

18.8.3 Recovery Command

To start the recovery process at an MTA, issue the following command:

```
RECOVER NODE "node-id" MTA WORKSPACE "workspace-name"
```

where *workspace-name* is the file specification of the peer MTA's workspace, for example:

Tru64 UNIX

```
/mnt/mta/workspace
```

♦

OpenVMS

```
node "SYSTEM password"::device:[MTA$node]
```

where *node* is the name of the node where the peer MTA is located, *password* is the password of the SYSTEM account on the peer MTA's node, and *device* is the device containing the peer MTA's workspace.

♦

When you issue the Recover command, the following warning text might be displayed: `Warning - Recovery initiated but the workspace is not locked`. This warning text means that the MTA is unable to lock the peer MTA's workspace. In this case, the restrictions relating to recovering a workspace that is not locked apply, see Section 18.8.1.

18.8.4 Recovery Finished Event

When an MTA stops copying messages from a peer MTA's workspace, it generates the Recovery Finished event. This event explains why the recovery process has stopped. Note that there is no counter for the Recovery Finished event.

The Recovery Finished event contains the following:

`Workspace = "workspace-name"`

where *workspace-name* is the file specification of the peer MTA's workspace.

This is followed by one of:

1 `Result = Recovery Complete`

2 `Result = Access to Workspace Lost`

3 `Result = Recovery Stopped by the MTA that Owns the Workspace`

Action

1 The recovery process stopped because there are no more messages in the peer MTA's workspace. The recovery of the peer MTA's workspace was successful and no action is required.

2 The recovery process stopped because the MTA lost access to the peer MTA's workspace. Reissue the Recover command. If the MTA is still unable to access the peer MTA's workspace at the time you reissue the Recover command, the Recover command fails. Check that the MTA has read, write, and delete access to the peer MTA's workspace and, if necessary, set up this access to the peer MTA's workspace.

3 The recovery process stopped because the peer MTA is being created. No further action is required.

18.8.5 Information Not Recovered About Messages

When an MTA recovers messages from a peer MTA's workspace, it does not copy any recorded information about those messages that might be held by the peer MTA, for example:

- Statistical information about messages recorded by the counters of the peer MTA's Agent, MTA, and Peer MTA entities.
- Accounting information
- Archived messages
- Message History information

You might be able to access the Accounting information and Archived messages held by the MTA whose messages were recovered. Accounting information can be examined using the Accounting Decoder tool. Similarly, Archived messages can be examined using the Message Decoder tool. You can run these tools from any MTA that can access the peer MTA's directories. See the appropriate appendix for information about running the Accounting Decoder and Message Decoder tools.

When an MTA copies a message from a peer MTA, the peer MTA does not log any Message History information about that message. Consequently, when the peer MTA is recreated it cannot provide Message History information about a message that was copied from its workspace. However, Message History information about a message copied from the peer MTA can be provided by the MTA that copied it.

19
Problems with Routing

This chapter describes the problems that might occur with routing messages and how to solve these problems. MTAs use information stored in the directory and in Agent and Peer MTA entities to route messages. If any of this information is incorrect, then an MTA may be unable to send a message to its intended recipient(s).

19.1 Discrepancies in Routing Information

Discrepancies between entries in the directory are detected by an MTA when it is trying to obtain routing information from the directory. If an MTA finds a discrepancy in the directory it generates the Directory Configuration Error event; see Section 19.3.4.

If you replicate routing information, inconsistencies can occur between routing information held by different DSAs in the same routing domain. If you replicate the shared routing information for the routing domain, the MTAs in your routing domain could be using a DSA that holds a shadow copy of it. You are advised to make any changes to the routing information at the master DSA. The master DSA is the DSA that holds the Naming Context for your routing domain. After you have modified the routing information at the master DSA you must update the shadow copies held by other DSAs. Until all the MTAs in your routing domain have the same routing information available to them, messages could be routed incorrectly.

Inconsistencies between routing information held in different routing domains could cause a message to take a circular route. When an MTA detects that a message is taking a circular route, that is, traveling along the same route more than once, the MTA generates the Loop Detected event; see Section 19.3.5.

Discrepancies can also occur between routing information in the directory and routing information held locally by an MTA in its Agent or Peer MTA entities. If an MTA is unable to find an Agent or Peer MTA entity that is named within a routing instruction held in the directory, then it generates one of the following events:

- Rejected Agent Connection (see Section 19.3.1)
- Unknown Agent (see Section 19.3.2)
- Unknown Peer Domain (see Section 19.3.3)

If you believe the MTA has incorrectly routed a message, follow the procedure described in Section 19.2.

19.2 Finding an O/R Address Entry in the Directory

If the routing information in the directory is incorrect or missing, then a message is either incorrectly routed or not delivered. To find out whether the routing information exists in the directory, check that the recipient's O/R address on the message has a corresponding entry in the directory.

If you have replicated the routing information, first check that the correct routing information is held by the master DSA for your routing domain. If the master DSA has the routing information, check that the same routing information exists at all the DSAs that hold shadow copies.

Use the MTS module to access and modify the information in the directory. See the *MTS Module Online Help* for information about creating MTS module entities and modifying their attributes.

Use the following command to display an O/R address entry in the directory:

```
SHOW MTS "/MTS=routing-domain-name" ORADDRESS "oraddress" -
ALL ATTRIBUTES
```

where:

- *routing-domain-name* is the name of your routing domain
- *oraddress* is an O/R address

For example, a message is addressed as follows:

```
Recipient = "C=NZ;A=NZ-PTT;P=ACME;OU1=WELL;CN=CLARE ROBERTS"
```

This message is not delivered to Clare Roberts but is delivered to another recipient. To find out if the routing information in the directory is either incorrect or missing display the O/R address entry in the directory using the following command:

```
SHOW MTS "/MTS=ACME" ORADDRESS -
"C=NZ;A=NZ-PTT;P=ACME;OU1=WELL;CN=CLARE ROBERTS" ALL ATTRIBUTES
```

If this command fails to return a matching O/R address entry, then remove the last term from the O/R address and try the command again, for example:

```
SHOW MTS "/MTS=ACME" ORADDRESS "C=NZ;A=NZ-PTT;P=ACME;OU1=WELL" -
ALL ATTRIBUTES
```

If this command does not return a matching O/R address entry, then remove the last term from the O/R address and issue the command again.

When the command returns a matching O/R address, you have found the O/R address entry in the directory that contains the closest match to the recipient's full O/R address. This O/R address entry should contain routing information as this is the entry that the MTA used when it routed the message.

Examine the routing information contained in the O/R address entry that you have found. The routing information identifies either an MTA entry or a Domain entry in the directory (or another O/R address entry if the recipient's messages are being redirected). Check that the entity named as a target in the routing instruction exists in the directory. If the entity does not exist, an appropriate event should have been generated (see Section 19.3.4). If the entity exists, then check that it contains the correct information.

If the information in the directory is incomplete or inaccurate, then modify the relevant entities of the MTS module to update the information in the directory.

19.3 Events Related to Problems with Routing

The following events relate to discrepancies or errors in the routing information held either in the directory or locally at the MTA in an Agent or Peer MTA entity:

- Rejected Agent Connection (Section 19.3.1)
- Unknown Agent (Section 19.3.2)
- Unknown Peer Domain (Section 19.3.3)

- Directory Configuration Error (Section 19.3.4)
- Loop Detected (Section 19.3.5)

Note

If you have replicated routing information, always check the routing information held at the master DSA when solving routing errors identified by these events. If the routing information held at the master DSA is correct, check that the shadow copies have been updated.

19.3.1 Rejected Agent Connection

This event occurs whenever an MTA rejects a connection from an Agent.

Note

This event only applies to Agents that use the XAPI interface.

This event is counted by the MTA entity's Rejected Agent Connections counter.

The information provided by the event depends on whether the Agent is a registered Agent or an unregistered Agent of the MTA.

For a registered Agent, the event contains the following:

```
Agent =
      [
      Agent Name = agent-name
      ]
```

where *agent-name* is the name of the registered Agent that the MTA is rejecting a connection from.

This is followed by one of:

1 `Reason Code = Unknown Agent`

2 `Reason Code = Invalid Password`

3 `Reason Code = Maximum Connections Exceeded`

For an unregistered Agent, the event contains the following:

```
Agent =
      [
      Mailbox Name = oraddress
      ]
```

where *oraddress* is the O/R address that the unregistered Agent supplied when it tried to connect to the MTA.

This is followed by one of:

4 `Reason Code = Unknown Agent`

5 `Reason Code = Invalid Password`

6 `Reason Code = Maximum Connections Exceeded`

Action

1 The MTA rejected the connection because it did not recognize the Agent's name. Agents are registered with the MTA as Agent entities. When the MTA receives a connection request from an Agent it searches for an Agent entity with the same name. If the MTA is unable to find an Agent entity with that name, it rejects the connection request.

Find out if the Agent named in the event should be registered. If it should be registered, then check that none of the MTA's Agent entities that represent Agents using the XAPI interface are incorrectly named. To do this, either check the create commands for Agent entities in the MTA's startup script or display the identifiers of the Agent entities using the following command:

```
SHOW NODE "node-id" MTA AGENT * TYPE
```

If you find an obvious error in the name of an Agent entity, delete the incorrectly named Agent entity and recreate it with the name specified in the event. Check that the MTA is exchanging messages with the Agent. Finally, correct the MTA's startup script.

If all the MTA's Agent entities are correctly named, create an Agent entity for the Agent named in the event. See Part II of *HP MAILbus 400 MTA Planning and Setup* for information about how to plan an Agent entity. If copies of the messages exchanged with the Agent are to be archived, then set up the Archive attribute (see Chapter 9). Include the commands that create, set up, and enable the Agent entity in the MTA's startup script.

2 The MTA rejected the connection because it did not recognize the Agent’s password.

Note that Agents do not have to provide a password when they make a connection request to the MTA. If a registered Agent of the MTA provides a password, then the MTA checks the password provided by the Agent with the password held by the Agent entity. If the passwords do not match, then the MTA rejects the connection request. If the Agent does not provide a password, then the MTA checks that there is no password held in the Agent entity. If there is a password in the Agent entity, then the MTA rejects the connection request.

Contact the person responsible for the Agent and verify the Agent’s password. Note that the Password attribute is a write-only characteristic attribute, so you cannot display its value.

If you have included the Agent entity’s Password attribute in the MTA’s startup script, then you can display the startup script and check the password.

If you have not included the Agent entity’s Password attribute in the MTA’s startup script, you are unable to check that the password is correct. Therefore, you have to assume that it has been incorrectly set up.

Use the following command to reset the Password attribute:

```
SET NODE "node-id" MTA AGENT "agent-name"  PASSWORD "password"
```

where *agent-name* is the identifier of the Agent entity, and *password* is the correct password.

It is recommended that you include the commands that create, set up, and enable this Agent entity in the MTA’s startup script.

3 The MTA rejected the connection because it had reached the limit of Agent connections that it can receive.

The Agent makes another attempt at connecting to the MTA, therefore no action is required. However, if this event occurs frequently, increase the value of the MTA entity’s Maximum Agent Connections attribute, see Section 7.3.1.

4 The MTA rejected the connection because it could not validate the O/R address supplied by the unregistered Agent. When an unregistered Agent connects to an MTA, it supplies the O/R address of the person using the Agent. The MTA validates this address by looking in the directory for a matching O/R address entry. If it cannot find one, then the MTA rejects the connection request.

If you want to check the O/R address entry in the directory against the O/R address supplied by the Agent, use the following command:

```
SHOW MTS "/MTS=routing-domain-name" ORADDRESS "oraddress" -
ALL ATTRIBUTES
```

where:

- *routing-domain-name* is the name of your routing domain
- *oraddress* is the O/R address provided by the Agent

If there is no corresponding O/R address entry in the directory, then the command returns an error. To find out how much of the O/R address has been entered in the directory, remove the last term from the O/R address and issue the command again as described in Section 19.2.

Inform the person responsible for managing the Agent about the discrepancy in the O/R addresses. Either the O/R address supplied by the Agent or the information in the directory needs to be changed. See the *MTS Module Online Help* for information about using the MTS module to modify information in the directory.

5 The MTA rejected the connection because it does not recognize the Agent's password. When an unregistered Agent connects to the MTA it supplies the O/R address of the person using the unregistered Agent and a password to the MTA. An O/R address entry in the directory can contain a password. The MTA validates the password supplied by the Agent against the password in the user's O/R address entry in the directory. If the password supplied by the Agent does not match the password held in the directory, then the MTA rejects the Agent's connection request. If the Agent does not provide a password, then the MTA checks that there is no password held in the user's O/R address entry in the directory. If there is a password in the user's O/R address entry, then the MTA rejects the connection request.

Contact the person responsible for managing the unregistered Agent and verify the password supplied by the Agent. The password held in the directory is a write-only characteristic attribute, so you cannot display its value. Therefore, you have to assume that it has been incorrectly set up. Reset the Password attribute in the user's O/R address entry using the ORaddress entity of the MTS module. See the *MTS Module Online Help* for information about modifying ORaddress entities.

6 The MTA rejected the connection because it has reached the limit of Agent connections that it can receive.

The Agent makes another attempt at connecting to the MTA, therefore, no action is required. However, if this event occurs frequently, you can increase the value of the MTA entity's Maximum Agent Connections attribute, see Section 7.3.1.

19.3.2 Unknown Agent

This event occurs whenever an MTA receives a message for delivery or export to an Agent that it does not recognize. Usually, this is because of a discrepancy in the routing information provided in the recipient's O/R address entry. This discrepancy could be due to one of the following:

- The name of the Agent in the routing information within the recipient's O/R address entry is incorrect.
- An Agent entity has been created with an incorrect name.
- No Agent entity has been created for the Agent named in the recipient's O/R address entry.
- The message has been transferred to the wrong MTA.

This event is counted by the MTA entity's Unknown Agents counter.

The event provides the following information:

```
A failed O/R Address from the MPDU = oraddress
Agent = agent-name
```

where *oraddress* is the recipient's O/R address and *agent-name* is the name of the Agent.

Action

Check that the routing information in the directory for the O/R address provided by the event is correct. If the routing information is incorrect, then use the ORaddress entity of the MTS module to modify the O/R address entry in the directory.

If the routing information in the directory is correct, then check that none of the MTA's Agent entities are incorrectly named. To do this, either check the create commands for Agent entities in the MTA's startup script or display the identifiers of Agent entities, using the following command:

```
SHOW NODE "node-id" MTA AGENT * ALL IDENTIFIERS
```

If you find an obvious error in the name of an Agent entity, delete the incorrectly named Agent entity and recreate it with the name specified in the event. Check that the MTA is exchanging messages with the Agent. Finally, correct the MTA's startup script.

If all the MTA's Agent entities are correctly named, create an Agent entity for the Agent named in the event. See Part II of *HP MAILbus 400 MTA Planning and Setup* for information about how to plan an Agent entity. If the Agent uses the XAPI interface and copies of messages exchanged with the Agent are to be archived, set up the Archive attribute (see Chapter 9). Include the commands that create, set up, and enable the Agent entity in the MTA's startup script.

19.3.3 Unknown Peer Domain

This event occurs when there is a discrepancy between the name of a Domain entry in the directory and the Peer Domain attribute of a Peer MTA entity.

This discrepancy prevents a boundary MTA transferring a message to a peer MTA in the routing domain represented by the Domain entry. This is because the boundary MTA is unable to find a Peer MTA entity for that peer MTA.

The discrepancy between the Domain entry and the Peer Domain attribute, also causes a boundary MTA to reject an association request from a peer MTA in another routing domain. This is because the boundary MTA is unable to find the Domain entry in the directory that represents the routing domain where the peer MTA is located. When the boundary MTA rejects the association request, it generates the Inbound Transfer Hard Rejection event in addition to this event.

This event can be due to one of the following errors:

- The directory does not contain a Domain entry that represents the routing domain where the peer MTA is located.
- No Peer MTA entity has been created to represent the peer MTA in the other routing domain.
- A Peer MTA entity has been created, but its Peer Domain attribute is incorrect.
- The message has been transferred to the wrong boundary MTA.

  This is because the routing instruction in a Domain entry is incorrect.

When this event is generated because the boundary MTA cannot find a Peer MTA entity, then this event contains the name of the Domain entry in the directory. When this event is generated because the boundary MTA rejects an association request from the peer MTA, then this event contains the value of the Peer Domain attribute.

This event is counted by the MTA entity's Unknown Peer Domains counter.

Action

Find out the correct name of the Domain entry in the directory. The correct name is the name that was planned. Check that there is a Domain entry in the directory with the correct name; using the following command:

```
SHOW MTS "/MTS=routing-domain-name" DOMAIN -
"domain-name" ALL ATTRIBUTES
```

where:

- *routing-domain-name* is the name of your routing domain
- *domain-name* is the correct name of the Domain entity that represents the peer MTA's routing domain

The action you take depends on whether or not the correct Domain entry exists in the directory.

- The Domain entry exists in the directory.

  Check that the MTA that generated the event is the boundary MTA that is connected to a peer MTA in the other routing domain.

  If the MTA that generated the event is not the correct boundary MTA, then you need to modify the routing instruction in the Domain entity so that it points to the correct boundary MTA. In addition, check that there is an MTA entry in the directory for the correct boundary MTA. See the *MTS Module Online Help* for information about displaying and modifying the Domain and MTA entities.

  If the MTA that generated the event is the correct boundary MTA, then the Peer MTA entity that represents the peer MTA is missing or has the incorrect value in its Peer Domain attribute. Find out if you have created a Peer MTA entity to hold information about a peer MTA in the other routing domain. If you have, then reset its Peer Domain attribute using the following command:

  ```
  SET NODE "node-id" MTA PEER MTA [TYPE = MANUALLY CONFIGURED, -
  NAME = "peer-mta-name"] PEER DOMAIN = "domain-name"
  ```

  where *peer-mta-name* is the name of the Peer MTA entity that holds information about the peer MTA in the other routing domain, and *domain-name* is the correct name of the Domain entry in the directory.

  If there is no Peer MTA entity, then you have to create one and set it up. See *HP MAILbus 400 MTA Planning and Setup* for information about how to plan and set up a Peer MTA entity.

- The Domain entry does not exist in the directory.

Create a Domain entry in the directory that represents the domain where the peer MTA is located. See *HP MAILbus 400 MTA Planning and Setup* for information about planning and setting up a Domain entity of the MTS module.

After you have created a Domain entry in the directory, make sure that the Peer Domain attribute of the Peer MTA entity contains the name of the Domain entry.

19.3.4 Directory Configuration Error

This event occurs whenever an MTA accesses the directory and the directory does not have all of the expected information. This is because there is an error in the way that the routing information within the directory has been set up.

This event is counted by the MTA entity's Directory Configuration Errors counter.

The event contains the following information:

```
Directory Operation = operation
MTS Entity = mts-id
Problem Entity Name = identifier
Problem Entity Type = type
Attribute Type = attribute
Error = error
```

If the MTA was directed to the entry containing the problem by another entry in the directory, then the identity of the referencing entry is also shown:

```
Referenced Entity Name = identifier
Referenced Entity Type = type
```

where:

- `operation` is the Directory Service operation that the MTA was attempting when the error occurred, for example, `Search`.
- `mts-id` is the name of the MTS entity.
- `identifier` is the name of an entity in the MTS module.
- `type` is the type of entity in the MTS module named in the previous bullet, for example, `ORaddress`.
- `attribute` is the attribute in the entity that contains the problem, if known.
- `error` is a description of the problem; for example, `Invalid Attribute Syntax`.

Action

The event identifies the MTS entity and the attribute that contains the error, and describes the problem. Find out what information is expected to be held in the entity.

Correct any errors by modifying the specified attribute or creating and modifying the entity that is causing the problem. See the *MTS Module Online Help* for information about modifying the attributes of entities belonging to the MTS module.

19.3.5 Loop Detected

This event occurs whenever an MTA detects a loop in the route taken by a message. A loop indicates that a message has been relayed through a particular X.400 management domain twice or has been handled by 50 MTAs.

An MTA can detect the following types of loop:

- External

  Each time a message enters an X.400 management domain the Global Domain Identifiers (GDIs) for that domain are added to the external trace information field in the message envelope. When an MTA receives a message it checks the GDIs in the external trace information field. An MTA detects an external loop when it receives a message that has previously been relayed to the X.400 management domain that the MTA is part of.

- Internal

  Each MTA that handles a message in the local routing domain adds its name to the internal trace information field in the message envelope. An MTA detects an internal loop when it receives a message that has been handled by 50 MTAs.

When an MTA detects a loop it does not deliver the message and sends a non-delivery report to the originator. Note that the non-delivery report is returned to the originator of the message only if the originator specifically requested a report or is able to receive reports. The MTA copies the message that contains the loop to the bad messages directory.

This event is counted by the MTA entity's Loops Detected counter.

The event contains one of the following:

1
```
Trace Type = External
Saved MPDU = file-spec
```

2 `Trace Type = Internal`
`Saved MPDU = `*`file-spec`*

where *`file-spec`* is the name of the file in the bad messages directory that contains the copy of the message.

Action

1 Use the Message Decoder tool to examine the message in the bad messages directory. For information about how to run the Message Decoder tool, refer to the appendix describing the operating system specific information.

Examine the Trace Information field within the saved copy of the MPDU to identify the loop. Each record of this trace contains a Global Domain Identifier (GDI) of an X.400 management domain through which the message has traveled.

You need to find out why the message was routed from your X.400 management domain and where the message went after it left your routing domain. This means that you have to look through the Trace Information and identify your routing domain. In order to identify your routing domain you need to know its GDIs. Note that a routing domain can have more than one GDI, but only one of its GDIs appears on the message. Use the following command to display the GDIs for your routing domain:

```
SHOW MTS "/MTS=routing-domain-name" GLOBAL DOMAIN -
IDENTIFIERS
```

where *`routing-domain-name`* is the name of your routing domain.

Look through the trace information in the saved message until you find a GDI for your routing domain. The GDI information following your routing domain's GDIs identifies the next X.400 management domain on the message's route. The loop could have begun when the message left your X.400 management domain.

To find out if the message should have been sent to this X.400 management domain, check the routing instruction for each recipient of the message.

The O/R address of each recipient is specified in the message envelope. Use the following command to check the routing instruction:

```
SHOW MTS "/MTS=routing-domain-name" ORaddress "oraddress" -
ROUTING INSTRUCTION
```

where:

- *`routing-domain-name`* is the name of your routing domain
- *`oraddress`* is the O/R address of a recipient on the message envelope

You need to make sure that the routing instruction for each recipient is correct and that none of the recipients are in your routing domain. If you find an error in the routing instruction for any of the recipients, then correct the error using the appropriate entities in the MTS module. See the *MTS Module Online Help* for information about modifying MTS module entities.

If you are unable to find an error in the routing instruction of any of the recipients, then do the following:

- Identify the boundary MTA that transferred the message from your X.400 management domain.
- Identify the Peer MTA entity that holds information about a peer MTA in the other X.400 management domain.
- Notify the person responsible for managing that peer MTA about the problem.

2 If an MTA in your routing domain detects a loop in a message that has not been outside your X.400 management domain, then the loop is likely to be due to discrepancies in the routing information held in the directory. Discrepancies can occur when information in the directory is being updated. Until the directory is fully and correctly updated, different MTAs in your routing domain could obtain different routing information from the directory. If you have replicated the routing information, updating the directory is not complete until all the DSAs that hold shadow copies of the directory are updated.

20 Problems with Resources

This chapter describes the problems that occur when an MTA cannot access a resource that it requires. The most important resources that the MTA needs are those provided by the operating system that it is running on and the directory. The System Interface Error (see Section 20.1) is generated by the MTA whenever it detects a problem with the operating system. The MTA generates the Directory Service Error (see Section 20.2) whenever it detects a problem with the directory service provided by the directory system agent (DSA).

20.1 System Interface Error

This event occurs whenever the operating system fails to do something that an MTA requested.

This event is counted by the MTA entity's System Interface Errors counter.

The event contains one of the following:

1
```
System Interface Error = Process Creation
Parameter =
Error Text = error-message
```

2
```
System Interface Error = Image Execution
Parameter = image
Error Text = error-message
```

3
```
System Interface Error = Interprocess Communication
Parameter =
Error Text = error-message
```

4
```
System Interface Error = File Access
Parameter = filename
Error Text = error-message
```

5
```
System Interface Error = Memory Allocation
Parameter =
Error Text = error-message
```

6
```
System Interface Error = Socket Operation
Parameter =
Error Text = error-message
```

7
```
System Interface Error = UID Operation
Parameter =
Error Text = error-message
```

8
```
System Interface Error = OpenVMS Management Operation
Parameter =
Error Text = error-message
```

where `image` is the name of an executable image, `filename` is the name of a file, and `error-message` is text provided by the operating system.

Action

1 The operating system is unable to create a process required by the MTA. This problem also affects other software running on the same node. Notify the person responsible for managing your system or see the documentation set applicable to your operating system.

2 The operating system was unable to run the named image.

Check that the image named in the event is in the correct directory and has the correct protection and ownership. See the appropriate appendix for the location, protection, and ownership of each image.

If you cannot find the program or image, then reinstall the MTA, see the MAILbus 400 MTA installation documentation.

3 The operating system is unable to provide communication between different processes.

Action is only required if this error affects the MTA so that it is no longer working correctly. If this happens, then notify your system manager or see the documentation set applicable to your operating system.

4 The operating system is unable to access a file.

Check that the file specified in the event has not been moved to a different directory or has the wrong file protection or ownership. For the location of the MTA's files and the correct protection and ownership for each file, see the appropriate appendix.

Tru64 UNIX

On Tru64 UNIX, if the error text is Too Many Files, in particular, when followed by a Forced Exit event, increase maxusers and rebuild the kernel.

maxusers is described in the Tru64 UNIX documentation as a Tru64 UNIX global keyword. Refer to the Tru64 UNIX documentation for information about this global keyword.
♦

5 The MTA has reached the limit of its allotted memory.

Notify the person responsible for managing your system or see the documentation set applicable to your operating system. If possible, allocate more memory to the MTA.

6 A socket is an end point for communication. The operating system has failed in an operation connected with a socket; for example, it is unable to create or delete a socket. Note that this problem can only occur on Tru64 UNIX systems. Notify the person responsible for managing your system or see the Tru64 UNIX documentation set.

7 The operating system has failed in an operation related to a unique identifier (UID); for example, it is unable to allocate a UID or is unable to convert a UID to a string format. The allocation of UIDs is done by DECnet. Notify the person responsible for managing your system or see the HP DECnet-Plus documentation and the documentation set applicable to your operating system.

8 The MTA failed when attempting a management operation; this is a temporary error. If the error persists, shut down and restart the MTA.

The System Interface Error event can occur *before* other events. Examine the event sink for any events with a time stamp later than the occurrence of this event and take the appropriate action to remedy any other problems.

If the event sink contains a Forced Exit event from this MTA, you can ignore any occurrences of the System Interface event that have a time stamp *later* than the Forced Exit event. This is because the operating system might not be responding as expected during the exit process.

20.2 Directory Service Error

There is a problem with the HP Enterprise Directory Service, specifically with the DSA. The DSA is a server that is responsible for storing the MTA's directory entries and providing access to them.

Directory Service Errors indicate that an MTA cannot obtain access to the directory or cannot obtain routing information from the directory.

This event is counted by the MTA entity's Directory Service Errors counter.

The event contains the following information:

```
Directory Operation = operation
DSA Address = address
MTS Entity = mts-name
Entity Name = identifier
Entity Type = type
Error = error
Error Type = error-type
```

where:

- `operation` is the directory service operation that the MTA was attempting when the error occurred; for example, `Bind`.
- `address` is the Presentation address of the DSA that the MTA is using or attempting to use.
- `mts-name` is the identifier of the MTS entity associated with the MTA.
- `identifier` is the identifier of the entity in the MTS module that relates to the directory information required by the MTA.
- `type` is the type of entity in the MTS module that relates to the directory information required by the MTA; for example `ORaddress`.
- `error` is a description of the directory service error, for example, `Busy`.
- `error-type` describes the category of directory service error; for example, `Service Problem`.

The following types of error can occur with the directory service:

1. `Error Type = Service Error`

   The DSA is unable to provide a service to the MTA.

2. `Error Type = Security Problem`

   The MTA is unable to access its routing information.

3. `Error Type = Communication Problem`

   The MTA is unable to communicate with the DSA.

4. `Error Type = Interface Problem`

   There is a problem with the interface between the MTA and the DSA.

Action

To solve problems with the HP Enterprise Directory Service you need to contact the person managing the DSA or see the HP Enterprise Directory Service documentation set. You can identify the DSA being used by the MTA from the DSA's Presentation address provided by this event. When you contact the person responsible for managing the DSA, supply all the relevant information provided in this event.

Some directory service problems reported by this event are also detected by the DSA, which generates a DSA event that gives more information about the problem. You could set up event dispatching so that you receive DSA events in addition to MTA events.

1 The following service errors can be reported by this event:

- `Error = Busy`

  No action is necessary as the MTA tries to reconnect to the DSA. However, if this event occurs frequently, contact the person managing the DSA and find out what is causing congestion at the DSA.

- `Error = Unavailable`

  Check the state of the DSA by issuing the following command at the node where the DSA is running:

  ```
  SHOW NODE "node-id" DSA STATE
  ```

  If the DSA does not exist or is in any state other than ON, contact the person managing the DSA and ensure that the DSA is running.

- `Error = Unwilling to Perform`

  This error is due to the way that the authentication characteristics in the DSA entity have been set up. The DSA does not recognize the MTA as an authorized directory user. Contact the person responsible for managing the DSA and ensure that the DSA entity is set up to allow the MTA access to the directory.

- `Error = DIT Error`

  One of the schema files that the MTA uses at the DSA, for example, MTS.SC or X400.SC, has been corrupted or is missing. Contact the person managing the DSA and have the schema file(s) reinstalled. Refer to the HP Enterprise Directory Service documentation set for the location of the schema files.

- `Error = Time Limit Exceeded`

The time limit that specifies how long a directory operation can take has expired. You can ignore this event as the MTA tries to access the directory again. However, if this event occurs frequently, contact the person managing the DSA and ensure that the DSA's time limit for directory operations is increased.

The following service errors occur when a DSA does not have the routing information required by the MTA and unsuccessfully attempts to refer to another DSA that does have the routing information. Referral by a DSA to another DSA is known as chaining:

- `Error = Chaining Required`
- `Error = Invalid Reference`
- `Error = Loop Detected`
- `Error = Out of Scope`
- `Error = Referral`

Start investigating chaining errors by checking that the MTA is using the correct DSA. The DSA's Presentation address provided by this event identifies the DSA being used by the MTA. If the MTA is not using the correct DSA, then the DUA defaults file needs to be reconfigured to contain the Presentation address of the DSA that holds the routing information for the MTA's routing domain. Consult with the managers of other applications on the MTA's node that also use the directory and find out if it is possible to reconfigure the DUA defaults file.

If the DSA identified by this event is the DSA that the MTA should be using, or if it is not possible to reconfigure the DUA defaults file, contact the person responsible for managing the DSA and ensure that chaining is avoided. To avoid chaining, the topography of the directory needs to be reorganized so that the routing information is replicated to the DSA identified in this event.

2 The following security errors can be reported by this event:

- `Error = Invalid Credentials`

  One or more of the MTA entity identifier, MTS entity identifier or MTA password are incorrect.

  Check that the value of each these identifiers or attribute stored in the directory is the same as the value in the MTA Startup script. It is possible that someone has modified the directory entries while the MTA is running.

When you have isolated the problem, make sure the directory entries and the MTA Startup script contain the same values and shut down and restart the MTA.

- `Error = Insufficient Access Rights`

  This error occurs when the controls that permit access to the directory have been modified to prevent the MTA accessing its routing information in the directory. In this case, the access controls have been set up at a naming context above the routing domain entry in the Directory Information Tree (DIT).

  To solve problems with access control, contact the person responsible for managing the DSA and ensure that the MTA can access the routing information in the appropriate MTS naming context.

3 Communication errors occur when there is a problem with the association between the MTA and the DSA. If a communication error occurs when the MTA is attempting to bind to the DSA, then the establishment of the association failed. If a communication error occurs when the MTA is attempting a Read or Search operation, then the association has been aborted. If an association between the MTA and the DSA fails or is aborted, the MTA attempts to reconnect to the DSA.

If communication errors occur frequently, contact the person managing the DSA and find out why the associations fail.

In the case of an establishment failure, before contacting the person responsible for managing the DSA, check the state of the DSA. If the DSA is in the ON state, check that the "DXD_CLNS" Transport Template exists. To do this, issue the following command at the node where the MTA that generated the event is running:

```
SHOW NODE "node-id" OSI TRANSPORT TEMPLATE "DXD_CLNS" ALL ATTRIBUTES
```

If the "DXD_CLNS" Transport Template entity does not exist, contact the person responsible for managing the DSA and ensure that the "DXD_CLNS" Template is created on the node where the MTA is running.

Note

If you try to create the MTA while the DSA is being created and enabled, if the MTA is unable to obtain access to the directory it generates the Directory Service Error event with the error `Transmit Error`. Ignore this event if it occurs under these conditions as the MTA tries to reconnect to the DSA.

However, if the MTA create command fails with the error `"Failed when communicating with the directory"`, all retries have failed, so check the state of the DSA. If the DSA is not running, contact the person responsible for managing the DSA and ensure that the DSA is in the ON state.

The following communication errors can be reported by this event:

- `Error = Abort Error`

  The association between the MTA and the DSA has been aborted. This could be due to the connection being lost because of network problems.

- `Error = Transmit Error`

  The MTA is unable to transmit connection information to the DSA.

- `Error = Receive Error`

  The MTA is unable to receive data from the DSA.

- `Error = Reject Error`

  The DSA has rejected attempts by the MTA to establish an association.

- `Error = Unexpected Event`

  The association between the MTA and the DSA has been aborted because of an error in one of the OSI lower layers.

- `Error = No DSA Address`

  The MTA does not have a Presentation address for the DSA or the Presentation address that the MTA is using for the DSA contains a syntax error. To solve this problem you need to reconfigure the DUA. See Part III of *HP MAILbus 400 MTA Planning and Setup* for information about configuring the DUA.

4 The following interface errors can be reported by this event:

- `Error = Memory`

  This is due to a temporary lack of memory at the MTA. When this error occurs, the MTA also generates the System Interface Error event with the error `Memory Allocation`. The System Interface Error event provides more information about the problem. Check the event sink for an occurrence of that event from this MTA and take the appropriate action as described in Section 20.1.

- `Error = Unknown`

  The cause of the problem is unknown. If this error occurs frequently contact HP. See Chapter 23 for information about how to contact HP and the information you need to supply.

- `Error = Decoding Problem`

  The MTA is unable to decode a response from the DSA. This is due to a software error. If this error occurs contact HP. See Chapter 23 for information about how to contact HP and the information you need to supply.

- `Error = Bad Identifier`

  The MTA cannot identify a reply it received from the DSA. This is due to a software error. If this error occurs contact HP. See Chapter 23 for information about how to contact HP and the information you need to supply.

21 Problems Collecting Information

This chapter describes the problems that can occur when collecting information about the functioning of an MTA or the messages that it has handled. Some resources that collect information, for example event dispatching, are set up as standard parts of MTA operation. Other resources, for example, Message History logging, Accounting and Archiving, can give rise to problems only when they are enabled.

21.1 Problems Collecting Events

You may find that you do not receive some or any of the events that you expect, although you know that there is a problem in your routing domain. Check with the person responsible for managing the Event Dispatcher to find out if the Event Dispatcher is operating normally.

If there is no problem with the Event Dispatcher, the problem could be one of the following:

- Event reporting has not been correctly set up on the network.
- Events are being sent to another event sink.
- Event filters have been set over-selectively so you do not see all the information you expect.
- The Event Dispatcher module has been disabled, deleted or is unavailable for some other reason, for example, because DECnet communication between the event stream and the event sink is not working properly.

Action

Make sure that event dispatching is set up correctly, as described in the HP DECnet-Plus documentation and in Chapter 13.

If the Event Dispatcher is unavailable, then once it is restored you need to manually restart MTA event dispatching. To do this, run the MTA's event dispatching script. Refer to the appendix describing the operating system specific information for the command to run this script.

21.2 Events Related to Problems with Collecting Information

The following events relate to failures in recording information about a message:

- Accounting Data Lost (Section 21.2.1)
- Message History Data Lost (Section 21.2.2)
- Archive Failed (Section 21.2.3)

21.2.1 Accounting Data Lost

This event occurs whenever an MTA is unable to log Accounting information about a message.

This event is counted by the Accounting Data Losses counter of the MTA entity or Peer MTA entity that generated the event.

The event contains the identifier of the message that was not logged.

Action

This problem is most likely to be caused because the disk that holds the MTA's Accounting files is full or has been corrupted. The System Interface Error event is also generated whenever there is a file access error. Check the event sink for an occurrence of the System Interface Error event from this MTA. The System Interface Error event provides additional information about the problem, see Section 20.1.

If the disk is full, take immediate action to delete any files that you do not need. If you do not want to delete any of the files on the disk, then copy some or all of the files to another medium. For the location of the MTA's Accounting directory and for information about how to run the Accounting Decoder tool, refer to the appendix describing the operating system specific information.

If this problem persists, tune the MTA by decreasing the value of the Accounting Purge Interval attribute so that the disk is purged more frequently, see Section 8.2.3.

21.2.2 Message History Data Lost

This event occurs whenever an MTA is unable to write History information about a message in the Message History workspace. Note that no history information is kept about reports or probes.

This event is counted by the MTA entity's Message History Data Losses counter.

The event contains the identifier of the message for which no history information was logged.

Action

This problem is most likely to be caused because the disk that holds the MTA's Message History workspace is full or has been corrupted. The System Interface Error event is also generated whenever there is a file access error. Check for an occurrence of the System Interface Error event from this MTA. The System Interface Error event provides additional information about the problem, see Section 20.1.

If the disk where the MTA's Message History workspace is located is full, take immediate action to make more space available on this disk. To immediately solve this problem, disable Message History logging at this MTA. Make sure that there is free space on the disk where the MTA's message history workspace is located before you re-enable Message History logging. For the location of the MTA's Message History workspace, refer to the appendix describing the operating system specific information.

Tune the MTA by decreasing the value of the Message History Purge Interval attribute so that the disk is purged more frequently, see Section 10.2.2.

21.2.3 Archive Failed

This event occurs whenever an MTA is unable to archive a message exchanged with an Agent or peer MTA.

This event is counted by the Failed Archives counter of the Agent entity or Peer MTA entity that generated the event.

The event contains the identifier of the message that was not archived.

Action

Since archived data is not purged automatically, this problem is most likely to be caused by the disk that holds the MTA's Archive files being full or having been corrupted. Check that your procedures for managing Archiving are working correctly.

The System Interface Error event is also generated whenever there is a file access error. Check for an occurrence of the System Interface Error event from this MTA. The System Interface Error event provides additional information about the problem, see Section 20.1.

If the disk is full and you do not want to delete any of the files on the disk, then copy some or all of the files to another medium. For the location of the MTA's Archive directory (Tru64 UNIX) or directories (OpenVMS), refer to the appendix describing the operating system specific information.

22 Software Problems

It is possible for problems to originate between the MTA software and other software, or in the MTA software itself. Section 22.1 describes how to solve problems related to the state of the MTA. Section 22.2 describes the events that report problems in the MTA software.

22.1 MTA Permanently in the Disabling or Enabling State

Once an MTA has been created, it is in one of the following states:

- ENABLING

  The enable command is being executed.
- ON

  The MTA is running.
- DISABLING

  The disable command is being executed.
- OFF

  The MTA is not running.

Use the following command to find out the state of an MTA:

```
SHOW NODE "node-id" MTA STATE
```

To change the state of an MTA from OFF to ON, issue the following command:

```
ENABLE NODE "node-id" MTA
```

When you issue this command, the state of the MTA immediately changes to ENABLING. When the command has fully executed, the state of the MTA changes to ON.

To change the state of an MTA from ON to OFF, issue the following command:

```
DISABLE NODE "node-id" MTA
```

When you issue this command, the state of the MTA immediately changes to DISABLING. When the command has fully executed, the state of the MTA changes to OFF.

An MTA should only be in the ENABLING or DISABLING state for a short period of time. However, if an MTA remains in one of these states, then the corresponding enable or disable command has failed to execute properly. To solve this problem you need to stop the MTA and then restart it as explained in the appendix describing the operating system specific information.

Note that only the MTA can be ON, ENABLING, DISABLING or OFF. Peer MTA and Agent entities are either ON or OFF.

22.2 Events Related to Problems with Software

The following events relate to problems with the MTA software:

- Internal Error (Section 22.2.1)
- Forced Exit (Section 22.2.2)

22.2.1 Internal Error

This event occurs whenever an MTA detects an error in its own software. This event is counted by the MTA entity's Internal Errors counter.

The event provides the following information:

```
Version = version
Diagnostic Number = diagnostic
```

where `version` is the version number of the MTA software and `diagnostic` is a reference number that identifies the error.

Action

Report all occurrences of this event to HP. See Chapter 23 for information about how to contact HP and the information you need to provide.

If an occurrence of this event is followed by a Forced Exit event, record all events that occur just before and after the Forced Exit event and any data associated with them. Try to restart the MTA by running the MTA startup procedure.

22.2.2 Forced Exit

This event occurs whenever an MTA is affected by an error due, for example, to a software fault in either the MTA or the operating system. Such faults prevent the MTA from continuing to function. Therefore, this event is likely to follow an Internal Error event or a System Interface Error event.

This event provides the following information:

`Exit Point =` *`component`*

where *`component`* is the component of the MTA that failed.

Action

If this event is preceded by a System Interface Error event, try to resolve the problem that caused the System Interface Error event (see Section 20.1). Try to restart the MTA by running the MTA startup procedure.

If this event is preceded by a System Interface Error event, and you cannot resolve the problem, or the event is preceded by an Internal Error event, then contact HP. See Chapter 23 for information about how to contact HP and the information to provide.

Note the Forced Exit event can lead to events that are generated as a result of the MTA deleting itself. You can ignore an occurrence of the System Interface Error event if it has a time stamp *later* than a Forced Exit event.

22.3 Events Related to License Problems

The Licensed Message Throughput Exceeded event occurs whenever an MTA detects that the message throughput has exceeded its license terms and agreement. This event is counted by the MTA entity's Licensed Message Throughput Exceeded counter.

The event provides the following information:

`Licensed Throughput=` *`messages_per_day`*.

where *`messages_per_day`* is the message throughput permitted by your license agreement.

Action

You must obtain a new license from HP that is appropriate for the message throughput that the MTA is now processing.

23 Reporting Your Problems

If you have a problem that you cannot solve by using the methods described in Part III you need to report your problem to HP. This chapter tells you how to contact HP and what evidence to gather before reporting your problem.

23.1 Contacting HP

Contact HP as follows:

- If you have a warranty or a service contract, contact your HP support center.
- If you do not have a warranty or a service contract, contact your local HP office to arrange for a service contract.

Contact your HP support center immediately on the occurrence of any problems which cannot be solved using this guide. Your HP Services representative is responsible for supporting your MAILbus 400 MTAs.

Alternatively, you can submit a Software Performance Report (SPR). SPR forms can be obtained from your local HP office. Report one problem per SPR and follow the instructions attached to the form. In addition to the information requested on the form, remember to include all the information relevant to your problem, as described in Section 23.2. When your SPR form is complete, submit the form and associated information to the SPR center nearest you (see the address list on the reverse side of the SPR instructions).

23.2 Gathering Information

Before contacting HP, collect as much information about the problem as you can. In your report to HP, always include the version number of the MTA software that you are running. The version number of the MTA software is held in the Version attribute of the MTA entity. Use the following command to display this attribute:

```
SHOW NODE "node-id" MTA VERSION
```

If you are reporting a problem that is caused by a protocol or parameter error, make a trace of the protocol information and include it with your report. See Section 16.4 for information about how to record protocol information.

Protocol errors are reported specifically by the RTSE Protocol Violation and Lower Layer Protocol Violation events. The following events can also indicate protocol or parameter errors:

- Inbound Failure event
- Outbound Failure event
- Outbound Establishment Failure event

Always report occurrences of the Internal Error event. The information provided by the Internal Error event includes a diagnostic reference number. Include this information in your report. When the Internal Error event is followed by a Forced Exit event, the MTA might record the following information at the same time as it records these events:

Tru64 UNIX

On Tru64 UNIX systems, information which is recorded in the system error log: /usr/adm/syslog.dated/*time-stamp*/syslog.log ♦

OpenVMS

On OpenVMS systems, information which is recorded in SYS$COMMON:[MTA]MTA.ERR or SYS$COMMON:[MTA]MTA.OUT ♦

Make a copy of the entries that were entered by the MTA in your system error log at the time the Internal Error and the Forced Exit events occurred. Include a copy of these system error log entries when you report the problem to HP. If you are reporting a problem related to the way the MTA works or the MTA's use of the directory, supply a copy of the MTA startup and directory population scripts that you are using.

Because problems are often difficult to reproduce with a different system configuration, define as precisely as possible the state of your system when the problem occurred. For example, the following information about your system will be helpful:

- The hardware that the MTA is running on.
- The version number of the operating system you are running.
- The type of disk storage you are using, or any special hardware configuration.

- Details of the operating environment of the MTA, for example, the memory resources available on the system where the MTA is running.
- The version number and mandatory update level of DECnet/OSI that you are using.

A Standards Information

This appendix introduces the 1992 messaging standards and explains how the MAILbus 400 MTA conforms to these standards.

This appendix lists the standards and recommendations that are referenced in the MTA documentation and that you may wish to have available. This appendix also provides information about how to obtain these documents, either by listing a document's ISBN number or by providing a contact address.

A.1 Brief Overview of the 1992 MHS Standards

In 1984, the International Telegraph and Telephone Consultative Committee (CCITT) introduced the 1984 X.400 Series of Recommendations to support the implementation of a global messaging system. Since then, the CCITT, the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC) have worked in close collaboration to develop recommendations and standards to replace the 1984 recommendations.

In 1988, the CCITT issued the 1988 X.400 Series of Recommendations, which are defined in the *CCITT Blue Book Volume VIII Fascicle VIII.7*, and the ISO/IEC issued corresponding X.400 standards defined in the *International Standard ISO/IEC 10021*. In the MAILbus 400 MTA documentation, these recommendations and standards are referred to as the 1988 Message Handling System Standards, or **1988 MHS Standards**.

Since 1988, the CCITT and ISO/IEC have continually revised their work. In 1992, the CCITT have consolidated their revisions with the 1988 X.400 Series of Recommendations to form the 1992 editions of these recommendations. For the purpose of the MTA documentation, the 1988 CCITT X.400 Recommendations, International Standard ISO/IEC 10021, *and* the revisions to these recommendations and standards are collectively called the **1992 MHS Standards**. Any reference in the MTA documentation to the 1992 MHS Standards includes the 1988 MHS Standards, unless indicated otherwise.

Section A.1.1 contains a detailed list of the revision documents that, combined with the 1988 MHS Standards, make up the 1992 MHS Standards.

The 1984 X.400 recommendations are referred to as **1984 MHS Standards** in the MTA documentation. Note also that the term "X.400" is used in the MTA documentation to refer to both the 1984 and the 1992 MHS Standards. Any exceptions are specifically defined.

The MAILbus 400 MTA conforms to the 1992 MHS Standards. In places where the CCITT recommendations and the ISO standards differ, the MAILbus 400 MTA product implements the ISO standard.

The MAILbus 400 MTA is capable of interworking with messaging systems conforming to the 1984, 1988 or 1992 MHS Standards. The MAILbus 400 MTA also conforms as closely as possible to the relevant profiles being developed in Europe, the USA and South East Asia. See Appendix B and Appendix C for further information about the MAILbus 400 MTA's conformance to the 1992 MHS Standards and related profiles. For information on how the MTA interworks with 1984 messaging systems see Chapter 6.

A.1.1 The 1992 Revisions to the 1988 MHS Standards

The following revision documents, combined with the 1988 MHS Standards, make up the 1992 MHS Standards:

- Revision of the CCITT 1988 X.400 Series of Recommendations

  The MHS Implementor's Guide Version 10 of February 1993
- Revisions of individual parts of International Standard ISO/IEC 10021
  - Part 10021-1: Corrigenda 1, 2, 3, 4, 5, 6, and Amendment 2
  - Part 10021-2: Corrigenda 1, 2, 3, 4, 5, 6, 7, and Amendments 1, 2
  - Part 10021-4: Corrigenda 1, 2, 3, 4, 5, 6, 7, 8, and Amendment 1
  - Part 10021-5: Corrigenda 1, 2, 3, 4, 5, 6, 7
  - Part 10021-6: Corrigenda 1, 2, 3, 4, 5, 6, 7
  - Part 10021-7: Corrigenda 1, 2, 3, 4, 5, and Amendment 1

A.2 Listing of Individual Standards and Recommendations

This section lists the individual standards and recommendations relevant to the MAILbus 400 MTA.

CCITT Recommendations

Recommendations developed by the International Telegraph and Telephone Consultative Committee (CCITT):

- *CCITT Blue Book Volume VIII Fascicle VIII.7, Data Communication Networks - Message Handling Systems (Recommendations X.400-X.420)*

  ISBN number 92-61-03721-6

  This book is referred to as the CCITT 1988 X.400 Recommendations.

- The MHS Implementor's Guide Version 10 of February 1993

  For information about how to obtain this document contact your national standards body, such as the British Standards Institute (BSI) in the United Kingdom, or the American National Standards Institute (ANSI) in the U.S.A. For the name and address of your national standards body, write to the following address:

  U.N. Bookstall,
  United Nations Assembly Building,
  New York, NY 11017, U.S.A.

- *CCITT Red Book Volume VIII Fascicle VIII.7, Data Communication Networks - Message Handling Systems (Recommendations X.400-X.430)*

  ISBN number 92-61-02361-4

  This book is referred to as the CCITT 1984 X.400 Recommendations.

- From *CCITT Blue Book Volume VIII Fascicle VIII.4, Data Communication Networks - Open Systems Interconnection (OSI) - Model and Notation, Service Definition (Recommendations X.200-X.219)*

  ISBN number 92-61-03691-0

  – X.208 *Specification of Abstract Syntax Notation One (ASN.1)*

  – X.209 *Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*

  – X.214 *Transport Service Definition for Open Systems Interconnection for CCITT Applications*

  – X.215 *Session Service Definition for Open Systems Interconnection for CCITT Applications*

- X.216 *Presentation Service Definition for Open Systems Interconnection for CCITT Applications*
- X.217 *Association Control Service Definition for Open Systems Interconnection for CCITT Applications*
- X.218 *Reliable Transfer: Model and Service Definition*

- From *CCITT Blue Book Volume VIII Fascicle VIII.5, Data Communication Networks - Open Systems Interconnection (OSI) - Protocol Specifications, Conformance Testing (Recommendations X.220-X.290)*

  ISBN number 92-61-03701-1

  - X.224 *Transport Protocol Specification for Open Systems Interconnection for CCITT Applications*
  - X.225 *Session Protocol Specification for Open Systems Interconnection for CCITT Applications*
  - X.226 *Presentation Protocol Specification for Open Systems Interconnection for CCITT Applications*
  - X.227 *Association Protocol Specification for Open Systems Interconnection for CCITT Applications*
  - X.228 *Reliable Transfer: Protocol Specification*

- From *CCITT Blue Book Volume VII Fascicle VII.3, Terminal Equipment and Protocols for Telematic Services (Recommendations T.0-T.63)*

  ISBN number 92-61-03611-2

  - T.50 *International Alphabet No. 5*
  - T.61 *Character Repertoire and Coded Character Sets for the International Teletex Service*

ISO International Standards

Standards referred to as ISO/IEC are developed by the International Organization for Standardization (ISO) in collaboration with the International Electrotechnical Commission (IEC).

- ISO/IEC 10021 *Information Processing Systems - Text Communication - Message-Oriented Text Interchange Systems (MOTIS)*

  Revisions to individual parts of International Standard ISO/IEC 10021:

  - Part 10021-1: Corrigenda 1, 2, 3, 4, 5, 6, and Amendment 2
  - Part 10021-2: Corrigenda 1, 2, 3, 4, 5, 6, 7, and Amendments 1, 2
  - Part 10021-4: Corrigenda 1, 2, 3, 4, 5, 6, 7, 8, and Amendment 1

  - Part 10021-5: Corrigenda 1, 2, 3, 4, 5, 6, 7
  - Part 10021-6: Corrigenda 1, 2, 3, 4, 5, 6, 7
  - Part 10021-7: Corrigenda 1, 2, 3, 4, 5, and Amendment 1
- ISO 8072 *Information Processing Systems - Open Systems Interconnection - Transport Service Definition*
- ISO/IEC 8073 *Information Processing Systems - Open Systems Interconnection - Connection Oriented Transport Protocol Specification*
- ISO 8326 *Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Service Definition*
- ISO 8327 *Information Processing Systems - Open Systems Interconnection - Basic Connection Oriented Session Protocol Specification*
- ISO 8613 *Information Processing - Text and Office Systems - Office Document Architecture (ODA) and Interchange Format*
- ISO 8649 *Information Processing Systems - Open Systems Interconnection - Service Definition for the Association Control Service Element*
- ISO 8650 *Information Processing Systems - Open Systems Interconnection - Protocol Specification for the Association Control Service Element*
- ISO 8822 *Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Service Definition*
- ISO 8823 *Information Processing Systems - Open Systems Interconnection - Connection Oriented Presentation Protocol Specification*
- ISO 8824 *Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)*
- ISO 8825 *Information Processing Systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*
- ISO/IEC 9066-1 *Information Processing Systems - Text Communication - Reliable Transfer Part 1: Model and Service Definition*
- ISO/IEC 9066-2 *Information Processing Systems - Text Communication - Reliable Transfer Part 2: Protocol Specification*
- ISO/IEC ISP 10611 *Information Technology - International Standardized Profiles AMH1n - Message Handling Systems - Common Messaging*
- ISO/IEC ISP 12062 *Information Technology - International Standardized Profiles AMH2n - Message Handling Systems - Interpersonal Messaging*

You can obtain these standards from the national standards body in your own country, such as the British Standards Institute (BSI) in the United Kingdom, or the American National Standards Institute (ANSI) in the U.S.A. For the name and address of your national standards body, write to the following address:

U.N. Bookstall,
United Nations Assembly Building,
New York, NY 11017, U.S.A.

Others

- *Stable Implementation Agreements for Open Systems Interconnection Protocols based on the Proceedings of the OSE Implementors' Workshop*

  This document is revised in regular intervals; the versions referenced in the MTA documentation are:

  - Version 7, Edition 1, December 1993.

    This version is also known as *NIST Special Publication 500-214*.
  - Version 3, Edition 1, December 1989.
  - Version 1, Edition 1, December 1987.

  You can obtain these documents from:

  National Technical Information Service (NTIS)
  U.S. Department of Commerce
  5285 Port Royal Road
  Springfield, VA 22161, U.S.A.

- CEN/CENELEC European Prestandards

  The European Prestandards referenced in this document are:

  - ENV 41214
  - ENV 41201
  - ENV 41202
  - ENV 41510

  You can obtain these documents from your national standards body if you are located in Europe, or from the following address:

  European Workshop for Open Systems
  Rue de Stassart, 36
  1050 Brussels, Belgium

- United States Government Open Systems Interconnection Profiles (US GOSIP)

  The US GOSIP versions referenced in this document are:

  - Version 1, which is available as Federal Information Processing Standard (FIPS) Publication 146, of August 1988.
  - Version 2, which is available as Federal Information Processing Standard (FIPS) Publication 146-1, of April 1991.

  You can obtain these documents from:

  National Technical Information Service (NTIS)
  U.S. Department of Commerce
  5285 Port Royal Road
  Springfield, VA 22161, U.S.A.

- United Kingdom Government Open Systems Interconnection Profile (UK GOSIP), Version 4.1

  This document is available as:

  - A Supplier Set, ISBN number 0 11 330568 0
  - A Supplier Set, First Update, ISBN number 0 11 330608 3
  - A Procurement Set, ISBN number 0 11 330567 2

- RFC 1328 *X.400 1988 to 1984 Downgrading*

  You can obtain this document from:

  SRI International
  Network Information Systems Center
  333 Ravenswood Avenue EJ291
  Menlo Park, CA 94025, U.S.A.

B
X.400 Elements of Service and Extensions

You need more detailed knowledge of the 1992 MHS Standards to use the information in this appendix.

This appendix:

- Summarizes how the MAILbus 400 MTA supports the Message Transfer and Message Handling/Physical Delivery X.400 Elements of Service defined in CCITT Recommendation X.400 and International Standard ISO/IEC 10021.
- Summarizes how the MAILbus 400 MTA handles extensions that are present in a message.

B.1 Elements of Service

The X.400 Elements of Service define features, functions or capabilities of an X.400 MHS. They are classified either as basic Elements of Service or as optional user facilities. These classifications are primarily applicable to organizations operating an ADMD public service.

Basic Elements of Service are the inherent and constituent parts of the MTS service; they are always provided and available. Optional user facilities can be selected by the user, either for individual messages or for a certain period of time.

Optional user facilities are either essential or additional. Essential optional user facilities are made available to all MTS users. Additional optional user facilities can be made available for national use, and for international use based on bilateral agreement. The essential and additional optional user facilities are listed in Annex B of CCITT 1988 Recommendation X.400 (International Standard ISO/IEC 10021-1).

A short summary of the support of the X.400 Elements of Service by the MAILbus 400 MTA is given below. For a more detailed listing see Table B–1 and Table B–2.

Summary of Support

The support of the X.400 Elements of Service by the MAILbus 400 MTA is as follows:

- All basic Elements of Service are supported.
- All essential optional user facilities are supported.
- Most additional optional user facilities are supported.

Table B–1 summarizes the Message Transfer (MT) Elements of Service and Table B–2 summarizes the Message Handling/Physical Delivery (MH/PD) Elements of Service. For a detailed definition of the X.400 Elements of Service, see Annex B of CCITT 1988 Recommendation X.400 (International Standard ISO/IEC 10021-1).

The following abbreviations are used in the tables:

84 Std	In the 1984 X.400 Standards
88 Std	In the 1988 X.400 Standards
M (B)	Basic Element of Service (mandatory)
M	Essential optional user facility (mandatory)
O	Additional optional user facility (optional)
UA	User Agent
EMS	Express Mail Service
PDS	Physical Delivery Service

Table B–1 The MAILbus 400 MTA's Support of MT Elements of Service

MT Service	84 Std	88 Std	Supported	Additional Information
Access management	M (B)	M (B)	Yes	
Alternate recipient allowed	M	M	Yes	
Alternate recipient assignment	O	O	Yes	
Content confidentiality	N/A	O	Yes	Pass-through[1]
Content integrity	N/A	O	Yes	Pass-through[1]
Content type indication	M (B)	M (B)	Yes	
Converted indication	M (B)	M (B)	Yes	
Conversion prohibition	M	M	Yes	
Conversion prohibition in case of loss of information	N/A	O	Yes	
Deferred delivery	M	M	Yes	
Deferred delivery cancellation	M	M	Yes	
Delivery notification	N/A	M	Yes	
Delivery time stamp indication	M (B)	M (B)	Yes	
Designation of recipient by Directory name	N/A	O	No[2]	
Disclosure of other recipients	M	M	Yes	
DL expansion history indication	N/A	M	Yes	
DL expansion prohibited	N/A	O	Yes	
Explicit conversion	O	O	No[3]	
Grade of delivery selection	M	M	Yes	
Hold for delivery	O	O	No[4]	

[1]Support of pass-through of P1 envelope fields which supply security features between secure MTS users

[2]Nonsupport of O/R names containing only Directory names for originator and recipient addresses or distribution lists

[3]Nonsupport of originator-specified explicit conversion; however, requests for explicit conversion are passed through

[4]Nonsupport of recipient-specified requirements with regard to holding messages for delivery until the target User Agent becomes available. However, you can provide a similar service by using the MAILbus 400 Message Store.

(continued on next page)

Table B–1 (Cont.) The MAILbus 400 MTA's Support of MT Elements of Service

MT Service	84 Std	88 Std	Supported	Additional Information
Implicit conversion	O	O	Yes	
Latest delivery designation	N/A	O	Yes	
Message flow confidentiality	N/A	O	Yes	Pass-through[1]
Message identification	M (B)	M (B)	Yes	
Message origin authentication	N/A	O	Yes	By recipient only
Message security labeling	N/A	O	Yes	By recipient only
Message sequence integrity	N/A	O	Yes	Pass-through[1]
Multi-destination delivery	M	M	Yes	
Non-delivery notification	M (B)	M (B)	Yes	
Non-repudiation of delivery	N/A	O	Yes	Pass-through[1]
Non-repudiation of origin	N/A	O	Yes	Pass-through[1]
Non-repudiation of submission	N/A	O	No[5]	
Original encoded information types indication	M (B)	M (B)	Yes	
Originator requested alternate recipient	N/A	M	Yes	
Prevention of non-delivery notification	O	O	Yes	
Probe	N/A	M	Yes	
Probe origin authentication	N/A	O	No[5]	
Proof of delivery	N/A	O	Yes	Pass-through[1]
Proof of submission	N/A	O	No[5]	

[1]Support of pass-through of P1 envelope fields which supply security features between secure MTS users

[5]Nonsupport of security services which require the MTA to use security information submitted in messages

(continued on next page)

Table B–1 (Cont.) The MAILbus 400 MTA's Support of MT Elements of Service

MT Service	84 Std	88 Std	Supported	Additional Information
Redirection disallowed by originator	N/A	O	Yes	
Redirection of incoming messages	N/A	O	Yes	Requires manual setup in directory
Report origin authentication	N/A	O	No[5]	
Requested delivery method	N/A	M	Yes	Validation of request only
Restricted delivery	N/A	O	No[6]	
Return of content	O	O	Yes	
Secure access management	N/A	O	No[7]	
Submission time stamp indication	M (B)	M (B)	Yes	
Use of distribution list	N/A	O	Yes	
User/UA capabilities registration	M (B)	M (B)	Yes	Requires manual setup in directory

[5]Nonsupport of security services which require the MTA to use security information submitted in messages

[6]Nonsupport of recipient-specified User Agents and distribution lists, from which no messages are to be delivered

[7]Nonsupport of strong authentication of password for User Agent to MTA access, and MTA to MTA access

Table B–2 The MAILbus 400 MTA's Support of MH/PD Elements of Service

MH/PD Service	88 Std	Supported	Additional Info
Additional physical rendition	O	Yes[1]	
Basic physical rendition	M (B)	Yes[1]	
Counter collection	M	Yes[1]	
Counter collection with advice	O	Yes[1]	
Delivery via Bureaufax service	O	Yes[1]	
EMS/Special delivery (one or other)	M	Yes[1]	
Ordinary mail	M (B)	Yes[1]	
Physical delivery notification by MHS	O	Yes[1]	Dependent on support of transfer out reports by Access Unit
Physical delivery notification by PDS	O	N/A	
Physical forwarding allowed	M (B)	Yes[1]	
Physical forwarding prohibited	O	Yes[1]	
Registered mail	O	Yes[1]	
Registered mail to addressee in person	O	Yes[1]	
Request for forwarding address	O	Yes[1]	Dependent on support of transfer out reports by Access Unit
Undeliverable mail with return of physical message	M (B)	N/A	

[1]MTA supports pass-through of MT parameters which the PDS uses.

B.2 Extensions

The CCITT 1988 X.400 Series of Recommendations defines a mechanism for extending the functionality of an MTA. Most of the features that were introduced in the CCITT 1988 X.400 Series of Recommendations are defined using this mechanism. Features that are defined in this way are known as **extensions**.

Some of the extensions introduced in the 1988 X.400 recommendations are optional, so not all X.400 MTA implementations are guaranteed to support them. Furthermore, some implementors may introduce extensions that are defined in future versions of the recommendations or which they themselves have devised. This means that a User Agent or Gateway submitting a message cannot be sure that all MTAs through which the message passes support the features that are required by the message. X.400 therefore allows an application to mark extensions as being critical to the submission, delivery, or transfer of a message. (In this case, transfer refers to the processing that the MTA does to the message before passing it to a User Agent or to another MTA or MTS.)

When a message is submitted to the MTA, the Interface Region checks the message for the presence of any extensions marked as being critical for submission. If such an extension is present, and the extension is supported, the MAILbus 400 MTA takes the appropriate action. If the MAILbus 400 MTA does not support the extension, the Interface Region does not accept the message.

Before sending an MPDU to the next destination in its route, the MAILbus 400 MTA checks it for the presence of any extensions marked as being critical for transfer or critical for delivery. If such an extension is present, and the extension is supported, the MAILbus 400 MTA takes the appropriate action. If the MAILbus 400 MTA does not support the extension, it non-delivers the MPDU if it is a message or a probe, or discards it if it is a report.

Table B–3 lists the 1988 X.400 extensions and shows which the MAILbus 400 MTA supports.

Table B–3 The MAILbus 400 MTA's Support of 1988 X.400 Extensions

Extension	Supported
Recipient-reassignment-prohibited	Yes
Originator-requested-alternate-recipient	Yes
DL-expansion-prohibited	Yes
Conversion-with-loss-prohibited	Yes[1]
Latest-delivery-time	Yes
Requested-delivery-method	Yes
Physical-forwarding-prohibited	Yes
Physical-forwarding-address-request	Yes
Physical-delivery-modes	Yes
Registered-mail-type	Yes
Recipient-number-for-advice	Yes
Physical-rendition-attributes	Yes
Originator-return-address	Yes
Physical-delivery-report-request	Yes
Originator-certificate	Yes
Message-token	Yes[2] [3]
Content-confidentiality-algorithm-identifier	Yes
Content-integrity-check	Yes
Message-origin-authentication-check	Yes[2]
Message-security-label	Yes[2]
Proof-of-submission-request	No
Proof-of-delivery-request	Yes[4]
Content-correlator	Yes
Probe-origin-authentication-check	"Critical for transfer" not supported
Redirection-history	Yes

[1]If this extension is marked as critical for transfer, the MAILbus 400 MTA avoids using a converter whose Lossy attribute has a "True" value.

[2]The MAILbus 400 MTA does not do any checking, but relays the extension so that recipient checks can take place.

[3]The MAILbus 400 MTA can deliver only asymmetric tokens.

[4]The recipient User Agent returns the requested information in the Message Delivery result.

(continued on next page)

Table B–3 (Cont.) The MAILbus 400 MTA's Support of 1988 X.400 Extensions

Extension	Supported
DL-expansion-history	Yes
Physical-forwarding-address	Yes
Recipient-certificate	Yes[4]
Proof-of-delivery	Yes[4]
Originator-and-DL-expansion-history	Yes
Reporting-DL-name	Yes[5]
Reporting-MTA-certificate	Yes[5]
Report-origin-authentication-check	Yes[5]
Originating-MTA-certificate	No
Proof-of-submission	No
An unknown integer	No
An unknown object ID	No

[4]The recipient User Agent returns the requested information in the Message Delivery result.

[5]The MAILbus 400 MTA does not generate such an extension, but relays any provided by another MTA.

C
Conformance to Regional Profiles

The MAILbus 400 MTA conforms to the 1992 MHS Standards. These consist of the CCITT 1988 X.400 Series of Recommendations and International Standard ISO/IEC 10021 (the international base standards defined for message handling systems), as well as the following revision documents:

- The CCITT MHS Implementor's Guide Version 10 of February 1993
- Revisions to individual parts of International Standard ISO/IEC 10021:
  - Part 10021-1: Corrigenda 1, 2, 3, 4, 5, 6, and Amendment 2
  - Part 10021-2: Corrigenda 1, 2, 3, 4, 5, 6, 7, and Amendments 1, 2
  - Part 10021-4: Corrigenda 1, 2, 3, 4, 5, 6, 7, 8, and Amendment 1
  - Part 10021-5: Corrigenda 1, 2, 3, 4, 5, 6, 7
  - Part 10021-6: Corrigenda 1, 2, 3, 4, 5, 6, 7
  - Part 10021-7: Corrigenda 1, 2, 3, 4, 5, and Amendment 1

The MAILbus 400 MTA also conforms to the International Standardized Profile ISO/IEC ISP 10611 Common Messaging (AMH1n), and to the draft International Standardized Profile ISO/IEC ISP 12062 Interpersonal Messaging (AMH2n). Section C.1 in this appendix outlines the MAILbus 400 MTA's conformance to these international profiles.

In addition, the MAILbus 400 MTA conforms to a number of profiles developed by other, regional, standards bodies. This appendix outlines the MAILbus 400 MTA's conformance to the following regional profiles:

- The *Stable Implementation Agreements for Open Systems Interconnection Protocols, Version 7, Edition 1, December 1993*, set up by the Open Systems Environment (OSE) Implementor's Workshop (OIW). In the MTA documentation, these Agreements are referred to as the OIW Stable Implementation Agreements. See Section C.2.
- United States Government OSI Profiles (US GOSIP). See Section C.3.

- United Kingdom Government OSI Profile (UK GOSIP) Version 4.1. See Section C.4.
- The following European Prestandards developed by the Comité Européen de Normalisation (CEN)/Comité Européen de Normalisation Electrotechnique (CENELEC) and the European Telecommunications Standard Institute (ETSI):
  - ENV 41201
  - ENV 41202
  - ENV 41214

  See Section C.5 for information on these Prestandards.

To fully understand the terminology used in this appendix you should have the referenced profiles available. The tables in this appendix are reproductions of tables from these profiles. Therefore, terms and classifications in the tables and sections in this appendix are used as per definition in the respective profile. See Appendix A for information about how to obtain these profiles.

C.1 The International Standardized Profiles (ISPs)

The International Standardized Profiles (ISPs) for Message Handling Systems are produced by international cooperation between the following regional standards communities:

- The North American OSE Implementor's Workshop (OIW)
- The European Workshop for Open Systems (EWOS) in collaboration with the European Telecommunications Standards Institute (ETSI)
- The Asia-Oceania Workshop (AOW)

International Standardized Profiles are balloted by the national standards bodies and, if agreed, are accepted as international standards. Thus, these profiles represent the harmonized views of MHS expert groups worldwide.

C.1.1 How the MAILbus 400 MTA Conforms

The MAILbus 400 MTA conforms to profiles in the following ISO/IEC ISP standards:

- ISO/IEC ISP 10611 *Information Technology - International Standardized Profiles AMH1n - Message Handling Systems - Common Messaging* (see Section C.1.1.1)

- ISO/IEC ISP 12062 *Information Technology - International Standardized Profiles AMH2n - Message Handling Systems - Interpersonal Messaging* (see Section C.1.1.2)

For information about obtaining these ISO/IEC profiles see Appendix A.

C.1.1.1 Conformance to ISO/IEC ISP 10611

The MAILbus 400 MTA conforms to the AMH11 profile described in Parts 2 and 3 of ISO/IEC ISP 10611 Common Messaging:

- ISO/IEC ISP 10611-2 Specification of ROSE, RTSE, ACSE, Presentation and Session Protocols for Use by MHS
- ISO/IEC ISP 10611-3 Message Transfer (P1) (AMH11)

Annex A of ISO/IEC ISP 10611-3 contains the International Standardized Profile Implementation Conformance Statement (ISPICS) proforma that defines how implementations must conform to this profile. The tables in this section are extracts from this ISPICS proforma and show the MAILbus 400 MTA's conformance to each functional group (Table C–1) and the relevant application contexts (Table C–2).

Italicized text in these tables indicates conformance information supplied by HP about the MAILbus 400 MTA product.

Table C–1 MTA Conformance to ISO/IEC ISP 10611-3 (AMH11) ISPICS: Global Statement of Conformance and Profile Conformance

Question	Response	Comments
Global statement of conformance		
Are all mandatory base standards requirements implemented?	*Y*	
Statement of profile conformance		
Are all mandatory requirements of profile AMH111 implemented?	*Y*	
Are all mandatory requirements of profile AMH112 implemented?	*Y*	
Are all mandatory requirements of any of the following optional functional groups implemented?		
Security (SEC)	*Y*	class(es): *S0*
Physical Delivery (PD)	-	*Can implement a co-located PDAU using the MAILbus 400 API.*
Conversion (CV)	*Y*	*Implicit Conversion*
Redirection (RED)	*Y*	
Latest Delivery (LD)	*Y*	
Return of Contents (RoC)	*Y*	
Distribution List (DL)	*Y*	
Use of Directory (DIR)[1]	-	*Nonsupport of O/R names containing only Directory names for originator and recipient addresses or distribution lists.*
’84 Interworking (84IW)	*Y*	

[1]*The MAILbus 400 MTA makes significant use of the X.500 Directory for routing, recipient capability assessment and distribution list expansion. Full support of the* **Use of Directory** *functional group is planned for a future release.*

Table C–2 shows the MAILbus 400 MTA’s conformance to the application contexts defined in the ISPICS for ISO/IEC ISP 10611-3.

Italicized text in this table indicates conformance information supplied by HP about the MAILbus 400 MTA product.

Table C–2 MTA Conformance to ISO/IEC ISP 10611-3 (AMH11) ISPICS: Supported Application Contexts

	Base			
Application Context	**CCITT**	**ISO/IEC**	**Profile**	**Support**
mts-transfer	m	m	m	*Y*
mts-transfer-protocol	m	o	c1[1]	*Y*
mts-transfer-protocol-1984	m	o	c2[2]	*Y*

[1]c1 - if conformance to AMH112 is claimed then m else o
[2]c2 - if conformance to AMH112 or the 84 Interworking functional group is claimed then m else o

C.1.1.2 Conformance to ISO/IEC ISP 12062

The MAILbus 400 MTA conforms to the AMH22 profile in Part 3 of ISO/IEC ISP 12062 Interpersonal Messaging: ISO/IEC ISP 12062-3 IPM Requirements for Message Transfer (P1) (AMH22).

Table C–3 in this section is an extract from the ISPICS proforma in Annex A of ISO/IEC ISP 12062-2 and shows the MAILbus 400 MTA's profile conformance.

Table C–3 MTA Conformance to ISO/IEC ISP 12062-3 (AMH22) ISPICS: Statement of Profile Conformance

Question	Response	Comments
IPM Security (SEC)	*Y*	class(es): *SO*
IPM Physical Delivery (PD)	-	*Can implement a co-located PDAU using the MAILbus 400 API.*
IPM Conversion (CV)	*Y*	
IPM Redirection (RED)	*Y*	
IPM Latest Delivery (LD)	*Y*	
IPM Return of Contents (RoC)	*Y*	

(continued on next page)

Table C–3 (Cont.) MTA Conformance to ISO/IEC ISP 12062-3 (AMH22) ISPICS: Statement of Profile Conformance

Question	Response	Comments
IPM Use of Directory (DIR)[1]	-	*Nonsupport of O/R names containing only Directory names for originator and recipient addresses or distribution lists.*
IPM 84 Interworking (84IW)	*Y*	*See NOTE below.*

[1]*The MAILbus 400 MTA makes significant use of the X.500 Directory for routing, recipient capability assessment and distribution list expansion. Full support of the* **Use of Directory** *functional group is planned for a future release.*

Note

The MAILbus 400 MTA supports the additional recommended practices for 1984 interworking specified in Annex C of ISO/IEC ISP 12062-1. These recommended practices concern downgrading of the IPM content from content type 22 (Interpersonal Messaging 1988) to content type 2 (Interpersonal Messaging 1984), and are aligned with the procedures specified in Section 6.1.3 of this guide.

C.2 The OIW Stable Implementation Agreements

The OIW Stable Implementation Agreements are the proceeds of the OSE Implementor's Workshop, a community of OSI vendors and purchasers who meet quarterly in the U.S.A. Their aim is to produce agreements on the features of OSI implementations. The OIW Stable Implementation Agreements are revised regularly, and therefore it is important to name the exact edition of the document to which a reference is made.

The MAILbus 400 MTA conforms to Version 7 Edition 1 of the OIW Stable Implementation Agreements, which was issued in December 1993. All references in the following subsections concern this version of the OIW Stable Implementation Agreements. The requirements relevant for the MAILbus 400 MTA are those laid out in Parts 7, 8, 29 and 30 of these Agreements.

For information about how to obtain a copy of the OIW Stable Implementation Agreements see Appendix A.

C.2.1 How the MAILbus 400 MTA Conforms

The MAILbus 400 MTA can use any of the Application Contexts defined for 1988 message transfer systems (MTSs) and so can function as either a 1984 or 1988 MTA.

Conformance to 1984 MHS protocols is defined in Part 7 of the OIW Stable Implementation Agreements; see Section C.2.1.1 for information about the MAILbus 400 MTA's conformance.

Conformance to 1988 MHS protocols is defined in Part 8, 29 and 30 of the OIW Stable Implementation Agreements; see Section C.2.1.2 for information about the MAILbus 400 MTA's conformance.

C.2.1.1 Conformance to 1984 MHS Protocols

The MAILbus 400 MTA satisfies the conformance requirements specified in Part 7, Clause 10.2 of the OIW Stable Implementation Agreements as follows:

- The MAILbus 400 MTA conforms to the requirements specified in Part 7, Clauses 5, 6 and 7, thus allowing the MTA to operate within an ADMD, PRMD or relaying PRMD. This assumes that the user has configured the O/R addresses and routing information used by the MAILbus 400 MTA appropriately.
- For routing within a PRMD, the MAILbus 400 MTA can operate as a Class 3 MTA as defined in Part 7, Clause 7.3.3.1.

The following qualifications on the requirements stated in Part 7 of the OIW Stable Implementation Agreements apply to the MAILbus 400 MTA:

- The MAILbus 400 MTA is capable of resuming an activity on an existing concurrent OSI association. To prevent the MAILbus 400 MTA from resuming activities on existing concurrent OSI associations in order to satisfy the restriction stated in Part 7, Clause 5.4.3 k), the MAILbus 400 MTA must be configured so that the Maximum Parallel Transfer Associations attribute of the relevant Peer MTA entity is set to one.
- The MAILbus 400 MTA is capable of using all or none of the Forced Nondelivery Times specified in Part 7, Clause 6.9.2, Table 16. To ensure the MAILbus 400 MTA uses all of them, the MAILbus 400 MTA must be configured so that its MPDU Expiry intervals are set to these values.
- Not all of the validation checks described in Part 7, Clause 6.8 are performed; in particular, there is no check to avoid inconsistency between the GDI in the first TraceInformation and the GDI in the MPDUIdentifier.

- For intra-PRMD connections, the MAILbus 400 MTA supports the InternalTraceInformation field but employs a more complex loop suppression algorithm than that implied in Part 7, Clause 7.3.2.
- For intra-PRMD connections, the MAILbus 400 MTA ensures the uniqueness of the MPDUIdentifier by a method other than that described in Part 7, Clause 7.3.4. The MAILbus 400 MTA assigns a 32 character identifier which is automatically generated.
- The MAILbus 400 MTA supports both the Unidentified (BilaterallyDefined) and ODA bodypart definitions described in Part 7, Annex B.

C.2.1.2 Conformance to 1988 MHS Protocols

The MAILbus 400 MTA satisfies the conformance requirements specified in Part 8, Clause 5 of the OIW Stable Implementation Agreements. The MAILbus 400 MTA operates as an "MHS-88-MTA" entity as described in Table 1 of the above-mentioned Clause.

Note

The OIW Stable Implementation Agreements are based on conformance to the International Standardized Profiles (ISPs) for Message Handling (also described in Section C.1). The current texts for these ISPs have been made available for reference in Parts 29 and 30 of the OIW Stable Implementation Agreements.

In addition to the requirements specified in the International Standardized Profiles, the MAILbus 400 MTA satisfies all the additional regional requirements specified in Clauses 6 and 8 of the OIW Stable Implementation Agreements.

C.3 US GOSIP

A United States Government OSI Profile (US GOSIP) is the standard reference for all US federal government agencies when acquiring and operating communication systems or services that are to conform to OSI protocols.

US GOSIP requirements are based on the agreements reached at the OSE Implementor's Workshops, which are published in the OIW Stable Implementation Agreements (see Section C.2). Different US GOSIP versions reference different versions of the OIW Stable Implementation Agreements.

C.3.1 How the MAILbus 400 MTA Conforms

The MAILbus 400 MTA conforms to two US GOSIP versions:

- Version 1, known also as Federal Information Processing Standard (FIPS) Publication 146.

  In the sections relating to Message Handling Systems, US GOSIP Version 1 references Part 7 of the OIW Stable Implementation Agreements, Version 1, Edition 1, December 1987.

- Version 2, known also as FIPS Publication 146-1.

  In the sections relating to Message Handling Systems, US GOSIP Version 2 references Part 7 of the OIW Stable Implementation Agreements, Version 3, Edition 1, December 1989.

  In addition, the MAILbus 400 MTA conforms to the requirement laid out in Clause 5.3.2 of US GOSIP Version 2. This clause specifies that an MTA must support routing on particular O/R address attributes.

See Appendix A for information about how to obtain these documents.

C.4 UK GOSIP Version 4.1

The United Kingdom Government OSI Profile (UK GOSIP) is designed to assist UK government departments when purchasing products based on OSI standards.

This profile will eventually be superseded by a European procurement specification, the European Procurement Handbook for Open Systems (EPHOS), produced by the European Commission. However, at this time, EPHOS is still under development.

The MAILbus 400 MTA conforms to UK GOSIP Version 4.1. UK GOSIP is published in two different sets:

- The Procurement Set for Version 4.0, aimed at the purchaser of OSI products.
- The Supplier Set, aimed at the supplier of OSI products.

  This is available as Version 4.0, and Version 4.1 (an update to Version 4).

See Appendix A for information about how to obtain these documents.

C.4.1 How the MAILbus 400 MTA Conforms

The MAILbus 400 MTA can use any of the Application Contexts defined for 1988 Message Transfer Systems and so can function as either a 1984 or 1988 MTA:

- Conformance to 1984 MHS protocols is defined in Chapter 1 of the UK GOSIP Version 4.1 Supplier Handbook, Volume 4 - Application Services (2).

  See Section C.4.1.1 for information about the MAILbus 400 MTA's conformance to the 1984 MHS protocols.

- Conformance to 1988 MHS protocols is defined in Chapter 2 of the UK GOSIP Version 4.1 Supplier Handbook, Volume 4 - Application Services (2).

  See Section C.4.1.2 for information about the MAILbus 400 MTA's conformance to the 1988 MHS protocols.

C.4.1.1 Conformance to 1984 MHS Protocols

Chapter 3, Appendix A of the UK GOSIP Version 4, Procurement Handbook Volume 3 - Application Services contains the procurement Protocol Implementation Conformance Statement (PICS) proforma to be filled in by suppliers of 1984 X.400 applications. This section refers to this appendix as the UK GOSIP MHS(84) procurement PICS appendix.

The tables in this section are reproductions of tables in the UK GOSIP MHS(84) procurement PICS appendix; only those tables are included that list conformance requirements relevant to the MAILbus 400 MTA.

- Table C–4 is a reproduction of a table in Section A.1 of the UK GOSIP MHS(84) procurement PICS appendix. It lists the required support of the MHS services. The MAILbus 400 MTA's conformance, where applicable, is indicated with *X*.
- Table C–5 is a reproduction of a table in Section A.2 of the UK GOSIP MHS(84) procurement PICS appendix. It lists the required support of MTA capabilities. The MAILbus 400 MTA's conformance, where applicable, is indicated with *X*.
- Table C–6 is a reproduction of a table in Section A.4 of the UK GOSIP MHS(84) procurement PICS appendix. It lists the required support of O/R name forms. The MAILbus 400 MTA's conformance, where applicable, is indicated with *X*.

Abbreviations used in this table:

C = Conditional
M = Mandatory

O = Optional

Italicized text in these tables indicates conformance information supplied by HP about the MAILbus 400 MTA product.

Table C–4 MTA Conformance to Service Support in UK GOSIP MHS(84) Procurement PICS Appendix

Service	Support		Notes
Message Transfer Service	M	[*X*]	
Interpersonal Messaging Service	C []	[ ]	Required unless only the GOSIP MHS(84) MTS conformance level is claimed

Table C–5 shows the MAILbus 400 MTA's conformance to the required MTA capability regarding origination, reception and relaying.

Table C–5 MTA Conformance to Required MTA Capability in UK GOSIP MHS(84) Procurement PICS Appendix

Capability	Support	
Origination	M	[*X*]
Reception	M	[*X*]
Relaying	O []	[*X*]

Table C–6 shows the MAILbus 400 MTA's support of the required O/R name forms.

Table C–6 MTA Conformance to O/R Name Form Support in UK GOSIP MHS(84) Procurement PICS Appendix

O/R Name Form/Variant	Support	
Form 1 Variant 1	M	[*X*]
Form 1 Variant 2	-	[*X*]
Form 1 Variant 3	-	[*X*]
Form 2	-	[*X*]

The MAILbus 400 MTA conforms to the requirements specified in the remaining relevant tables of the UK GOSIP MHS(84) procurement PICS appendix as follows:

- The MAILbus 400 MTA supports all of the MT service elements listed in Section A.6 of the UK GOSIP MHS(84) procurement PICS appendix for origination, reception and relay. See also Appendix B in this guide.
- The MAILbus 400 MTA supports all of the O/R name attributes listed in Section A.7 of the UK GOSIP MHS(84) procurement PICS appendix for origination and reception.
- The MAILbus 400 MTA supports all of the O/R name attributes listed in Section A.8.1 of the UK GOSIP MHS(84) procurement PICS appendix for inter-domain relay, intra-domain relay and intra-domain delivery, except for domain-defined attributes. No particular attributes are required for delivery.

C.4.1.2 Conformance to 1988 MHS Protocols

Chapter 3, Appendix B of the UK GOSIP Version 4, Procurement Handbook Volume 3 - Application Services contains the procurement Protocol Implementation Conformance Statement (PICS) proforma to be filled in by suppliers of OSI Message Handling applications. This section refers to this appendix as the UK GOSIP MHS(88) procurement PICS appendix.

The tables in this section are reproductions of tables in the UK GOSIP MHS(88) procurement PICS appendix; only those tables are included that list conformance requirements relevant to the MAILbus 400 MTA.

- Table C–7 is a reproduction of a table in Section B.1 of the UK GOSIP MHS(88) procurement PICS appendix. It lists the required support of the system and service configuration. The MAILbus 400 MTA's conformance, where applicable, is indicated with *X*.
- Table C–8 is a reproduction of a table in Section B.2 of the UK GOSIP MHS(88) procurement PICS appendix. It lists the required support of common messaging functional groups. The MAILbus 400 MTA's conformance, where applicable, is indicated with *X*.

Abbreviations used in this table:

C = Conditional
M = Mandatory
O = Optional

Italicized text in these tables indicates conformance information supplied by HP about the MAILbus 400 MTA product.

Table C–7 MTA Conformance to System/Service Configuration in UK GOSIP MHS(88) Procurement PICS Appendix

System/Service Configuration	Support	Notes
UA	O [ ] [ ]	P7 support is required
MTA	O [ ] [*X*]	P1[1] support is required
Combined MTA-UA	O [ ] [ ]	P1 support is required P7 support is not applicable
Combined MTA-MS	O [ ] [ ]	P1 support is required P7 support is required
Combined MTA-MS-UA	O [ ] [ ]	P1 support is required
P7 support	O [ ] [ ]	P7 support is operational for remote UA access
MHS service[1] remote access to UA	O [ ] [ ]	Using an unspecified protocol - treat as combined MTA-UA
P7 access	O [ ] [ ]	Treat as combined MTA-MS with P7 support
P1 access	O [ ] [*X*]	Treat as MTA with P1 support only

[1] An MHS ‘service’ in this context is one which is offered for remote access and where ownership of the internal system components remains with the service provider.

The MAILbus 400 Message Transfer Agent offers a Message Application and Message Transfer API (the MAILbus 400 Application Program Interface) according to the X/Open™ X.400 API Specifications. Thus the MAILbus 400 MTA will not normally be used simply as a relay-only MTA as implied by this table.

Table C–8 MTA Conformance to Common Messaging Functional Groups in UK GOSIP MHS(88) Procurement PICS Appendix

Functional Group	Support			Notes
	MTA	MS	UA	
MTA Kernel	M [*X*]	-	-	
MTS Kernel	C [ ]	-	M [ ]	Required for an MTA unless only P1 support is proposed
MS Kernel	-	M [ ]	C [ ]	Required for a UA with P7 support
84 Interworking	O [ ][*X*]	-	-	
DL Expansion[1]	O [ ][ ]	-	-	
Use of Directory[2]	O [ ][ ]	-	O [ ][ ]	
Secure Messaging	O [ ][*X*]	O [ ][ ]	O [ ][ ]	
security class0	O [ ][*X*]	O [ ][ ]	O [ ][ ]	
security class1	O [ ][ ]	O [ ][ ]	O [ ][ ]	
security class2	O [ ][ ]	O [ ][ ]	O [ ][ ]	
confidential variant	O [ ][ ]	O [ ][ ]	O [ ][ ]	
Physical Delivery	O [ ][*X*][3]	-	O [ ][ ]	Not applicable to an MTA with only P1 support

[1]*The MAILbus 400 MTA supports all of the Elements of Service defined in the CCITT 1988 X.400 Recommendations that are associated with DL Expansion, but it does not support the three DL policies described in the UK GOSIP specification.*

[2]*The MAILbus 400 MTA makes significant use of the X.500 Directory for routing, recipient capability assessment and distribution list expansion. Full support of the* **Use of Directory** *functional group is planned for a future release.*

[3]*The MAILbus 400 MTA's support and use of the X/Open X.400 Application Program Interface allows for support of a physical delivery access unit (PDAU) conforming to the* **Physical Delivery** *functional group.*

C.5 CEN/CENELEC European Prestandards (ENVs)

The Comité Européen de Normalisation (CEN)/Comité Européen de Normalisation Electrotechnique (CENELEC) and the European Telecommunications Standard Institute (ETSI) are standards bodies responsible for issuing standards on a European basis.

The European standards concerned with Message Handling Systems are produced by collaboration between MHS experts in the EWOS and ETSI committees.

C.5.1 How the MAILbus 400 MTA Conforms

The CEN/CENELEC committees and the European Telecommunications Standard Institute (ETSI) are responsible for the publication of M-IT-02, a taxonomy of profiles for the development of European Standards. M-IT-02 provides for several profiles for Message Handling, which are issued as European Prestandards (ENV). The profiles that the MAILbus 400 MTA conforms to, their corresponding ENV numbers and the Message Transfer protocols that they define are listed in the following table.

Profile No.	ENV No	Title	Message Transfer Protocol Defined	Date of Issue
A/3211	ENV 41201[1]	*Private Message Handling System: - UA and MTA; Private Management Domain to Private Management Domain*	1984 P1 and P2	February 1988
A/311	ENV 41202[1]	*Message Handling Systems: User Agent (UA) plus Message Transfer Agent (MTA): Access to an Administration Domain (ADMD)*	1984 P1 and P2	August 1987
A/MH11	ENV 41214	*Application Profile for Message Handling 11: Message Handling Systems - Common Facilities - MTA and MTS*	1988 P1	May 1992

[1]Amendments to this profile have been produced in January 1994 to enable implementations conforming to the 1984 Recommendations to interwork with 1988 implementations. These amendments are considered to be part of these ENVs when considering the MAILbus 400 MTA's conformance.

See Section C.5.1.1 for information about the MAILbus 400 MTA's conformance to ENV 41201.

See Section C.5.1.2 for information about the MAILbus 400 MTA's conformance to ENV 41202.

See Section C.5.1.3 for information about the MAILbus 400 MTA's conformance to ENV 41214.

For information about how to obtain a copy of these ENVs see Appendix A.

C.5.1.1 Conformance to ENV 41201 of February 1988

Conformance to ENV 41201 is a requirement of the 1984 Interworking Functional Group of ENV 41214 (see Section C.5.1.3). Note that the MAILbus 400 MTA conforms to Clause 6.8.1 of ENV 41214 and so will generate the application protocol identifier value of 1 (not 8883) in RTS connect requests.

The following qualifications on the requirements stated in ENV 41201 apply to the MAILbus 400 MTA:

- The MAILbus 400 MTA is capable of resuming an activity on an existing concurrent OSI association. To prevent the MAILbus 400 MTA from resuming activities on existing concurrent OSI associations in order to satisfy the restriction stated in Clause 2.4, the MAILbus 400 MTA must be configured so that the Maximum Parallel Transfer Associations attribute of the relevant Peer MTA entity is set to one.
- The MAILbus 400 MTA is capable of using Transport expedited data for Transport Classes other than Class 0. To prevent the use of Transport expedited data in order to satisfy the restriction stated in Clause 2.6.8, the relevant Peer MTA entity of the MAILbus 400 MTA must be configured to use a Transport template with this feature switched off.
- The MAILbus 400 MTA supports the 1984 P1 Internal Trace Information field defined in the OIW Stable Implementation Agreements (Clause 7.3.2) when interworking with 1984 MTAs in the same CCITT management domain. To prevent the use of the internal trace information in order to satisfy the restriction stated in Table 4 of ENV 41201, configure the MAILbus 400 MTA so that it is always in a different CCITT management domain to 1984 peer MTAs.
- The MAILbus 400 MTA supports the ISO 6937 IPM bodypart which is defined in ISO 9065 and referenced in Clause 5.4 and Table 4 of ENV 41201.

C.5.1.2 Conformance to ENV 41202 of August 1987

Conformance to ENV 41202 is a requirement of the 1984 Interworking Functional Group of ENV 41214 (see Section C.5.1.3). Note that the MAILbus 400 MTA conforms to Clause 6.8.1 of ENV 41214 and so will generate the application protocol identifier value of 1 (not 8883) in RTS connect requests.

The qualifications regarding the MAILbus 400 MTA's conformance are the same as those listed for ENV 41201 in Section C.5.1.1.

C.5.1.3 Conformance to ENV 41214 of May 1992

ENV 41214 is the first completed European profile concerned with 1988 MHS functionality.

Annex H of ENV 41214 is a Protocol Implementation Conformance Statement (PICS) proforma annex that contains profile-specific tables indicating how an implementation should conform to the major features of the A/MH11 profile. The following tables are reproductions of the tables in the PICS proforma annex and show how the MAILbus 400 MTA conforms to ENV 41214:

- Table C–9 in this appendix is a reproduction of Table H-1 in ENV 41214 and shows the MAILbus 400 MTA's conformance to the application contexts specified in ENV 41214.
- Table C–10 in this appendix is a reproduction of Table H-2 in ENV 41214 and shows the MAILbus 400 MTA's conformance to the abstract operations specified in ENV 41214.
- Table C–11 in this appendix is a reproduction of Table H-3 in ENV 41214 and shows the MAILbus 400 MTA's conformance to the functional groups specified in ENV 41214.

Abbreviations used in these tables:

C = Conditional
M = Mandatory
O = Optional

Italicized text in these tables indicates conformance information supplied by HP about the MAILbus 400 MTA product.

Table C–9 MTA Conformance to Application Contexts in ENV 41214

Application Contexts	A/MH11	Implemented
A/MH11.1	O	*Yes*
A/MH11.2	O	*Yes*

Table C–10 MTA Conformance to Abstract Operations in ENV 41214

Abstract Operations	A/MH11	Implemented
Message Transfer (MTSE)	M	*Yes*
Message Submission (MSSE)	O	*No*[2]
Message Delivery (MDSE)	O	*No*[2]
Message Administration (MASE)	C[1]	*No*[3]

[1]Mandatory if MSSE or MDSE is implemented.

[2]*Submission and Delivery services are implemented according to the X/Open X.400 API Specification, which offers most, but not all, of these P3 (MTS Access Protocol) services.*

[3]*Administration services are implemented using the directory; the MTA offers some, but not all, of these P3 (MTS Access Protocol) services.*

Table C–11 MTA Conformance to Functional Groups in ENV 41214

Functional Groups	A/MH11	Implemented
Minimum Kernel	M	*Yes*
Extended Kernel	O	*No*[2]
Redirection	O	*Yes*
Distribution List	O	*Yes*
Conversion	O	*Yes*
Use of Directory	O	*No*[3]
Physical Delivery		
PD0	O	*Yes*
PD1	O	*No*
84 Interworking	C[1]	*Yes*
Security		
S0	O	*Yes*

[1]Mandatory in A/MH11.2

[2]*All Elements of Service (EOS) except the Hold for Delivery EOS are implemented.*

[3]*Nonsupport of O/R names containing only Directory names for originator and recipient addresses or distribution lists. The MAILbus 400 MTA makes significant use of the X.500 Directory for routing, recipient capability assessment and distribution list expansion. Full support of the* **Use of Directory** *functional group is planned for a future release.*

(continued on next page)

Table C–11 (Cont.) MTA Conformance to Functional Groups in ENV 41214

Functional Groups	A/MH11	Implemented
S0a	O	*No*
S1	O	*No*
S1a	O	*No*
S2	O	*No*
S2a	O	*No*

D
The Tru64 UNIX Implementation of the MTA

This appendix describes the following features that are specific to the Tru64 UNIX implementation of the HP MAILbus 400 Message Transfer Agent:

- The privileges that you need before you can manage a MAILbus 400 MTA (Section D.1)
- The directories used by a MAILbus 400 MTA (Section D.2)
- The files installed with a MAILbus 400 MTA (Section D.3)
- The tools supplied with the MAILbus 400 MTA server subset (Section D.4)
- How to manually execute scripts supplied with the MAILbus 400 MTA (Section D.5)
- How to specify a different port number for Agents using the API Server over TCP/IP, if required (Section D.6).
- How to stop and start a MAILbus 400 MTA (Section D.7)
- How to restart a MAILbus 400 MTA server that is not responding to management (Section D.8).

D.1 Privileges

To manage the MTA or run any of the tools provided with the MTA, you must use an account that has superuser privileges.

D.2 Directories Used by the MTA

During normal operation, the MTA places messages in several single directories. The directories that are usually accessed only by the MTA are in the MTA's workspace. The MTA's workspace is under /var/mta/workspace and directories, which you can access, are directly under /var/mta. See Table D–1 for a list of the MTA's directories and workspace.

Table D–1 The MTA's Directories

Directory	Description
/var/mta/accounting	Contains Accounting files
/var/mta/archive	Contains archived messages
/var/mta/bad_msgs	Contains MPDUs or IPM bodyparts that the MTA cannot process
/var/mta/trace	Contains trace binary files
/var/mta/workspace	Contains the MTA's workspace

D.3 Files Installed on Your System

The following tables describe the files installed with the MAILbus 400 MTA:

- Table D–2 lists the contents of the MAILbus 400 MTA Management subset.
- Table D–3 lists the contents of the MAILbus 400 MTA Server subset.
- Table D–4 lists the contents of the MAILbus 400 MTA Base subset.

Note that these tables do not show the physical location of the files as they are installed from the individual subsets, but show the softlinks from the installed files. The physical location of the MTA's installed files is as follows:

`/directory/opt/subsetname/pathname`

where

- `/directory` is either /var if the softlink is in /var, or /usr if the softlink is in /usr.
- `subsetname` is the name of the subset from which the file was installed. This is one of the following:

  MTAANETMAN*nnn* for the MAILbus 400 MTA Management subset
  MTAASRVR*nnn* for the MAILbus 400 MTA Server subset
  MTAABASE*nnn* for the MAILbus 400 MTA Base subset

  where *nnn* is the version number of the product.
- `/pathname` is the complete path name of the softlinked file, for example `/var/mts/scripts/populate_mts_example.ncl`

For example, the physical location of the file referenced by the softlink `/usr/sbin/mts_cml_script` is:
`/usr/opt/MTAANETMAN140/usr/sbin/mts_cml_script`

Table D–2 The Files Installed from a MAILbus 400 MTA Management Subset

Description	Directory	File	Ownership	Protection
MTS Image	/usr/sbin	mts	root	lrwsr-xr-x
MTS CML Script	/usr/sbin	mts_cml_script	root	lrwxr-xr-x
MTA Help	/usr/share/dna/dict	mta_module.hlp	root	lrw-r–r–
MTS Help	/usr/share/dna/dict	mts_module.hlp	root	lrw-r–r–
MTA MSL	/usr/share/dna	ncl_dna5_mta.ms	root	lrw-r–r–
MTS MSL	/usr/share/dna	ncl_dna5_mts.ms	root	lrw-r–r–
Example Directory Script	/var/mts/scripts	populate_mts_example.ncl	root	lrw-r–r–
Populate Countries Script	/var/mts/scripts	populate_countries.ncl	root	lrw-r–r–

Table D–3 The Files Installed from a MAILbus 400 MTA Server Subset

Description	Directory	File	Ownership	Protection
MTA CML Script	/usr/sbin/mta	mta_cml_script	root	lrwxr-xr-x
MTA Startup Script	/var/mta/scripts	start_mta.ncl	root	lrw——-
MTA Shutdown Script	/var/mta/scripts	stop_mta.ncl	root	lrw-r–r–
MTA Setup Script	/var/mta/scripts	mta_setup	root	lrwxr–r–
MTA Event Dispatching Script	/var/mta/scripts	start_mta_event_dispatching.ncl	root	lrw-r–r–
MTA CLNS Transport Template Script	/var/mta/scripts	create_mta_clns_templates.ncl	root	lrw-r–r–
MTA CONS Transport Template Script	/var/mta/scripts	create_mta_cons_templates.ncl	root	lrw-r–r–
MTA Create Externally Defined Bodypart Script	/var/mta/scripts	create_mta_extdef_bodyparts.ncl	root	lrw-r–r–
MTA Release Notes	/usr/doc	mailbus400_mta.release_notes	root	lrw-r–r–
VP Image	/usr/examples/mta	mtamail	root	lrwxr-xr-x
VP Script	/usr/examples/mta	mta_vp.sh	root	lrwxr–r–
MTA images	/usr/sbin/mta	mta	root	lrwsr-xr-x
		mta_irchild	root	lrwxr-xr-x
		mta_irserver	root	lrwxr-xr-x

(continued on next page)

Table D–3 (Cont.) The Files Installed from a MAILbus 400 MTA Server Subset

Description	Directory	File	Ownership	Protection
		mta_mpchild	root	lrwxr-xr-x
		mta_mpserver	root	lrwxr-xr-x
		mta_rlchild	root	lrwxr-xr-x
		mta_rlserver	root	lrwxr-xr-x
		mta_wjchild	root	lrwxr-xr-x
		mta_wjserver	root	lrwxr-xr-x
		mta_remote_api_server	root	lrwxr-xr-x
Accounting Decoder tool	/usr/sbin/mta	mta_accdecoder	root	lrwxr-xr-x
Message Decoder tool	/usr/sbin/mta	mta_decoder	root	lrwxr-xr-x
Converter images	/usr/sbin/mta /converters/ipm	generalt61tolatin1	root	lrwxr-xr-x
		decdxtoddif	root	lrwxr-xr-x
		decdxtolatin1	root	lrwxr-xr-x
		decmcstolatin1	root	lrwxr-xr-x
		externaldeftobilatdef	root	lrwxr-xr-x
		externaldeftoposte	root	lrwxr-xr-x
		generaltoia5	root	lrwxr-xr-x
		generaltot61	root	lrwxr-xr-x
		ia5tolatin1	root	lrwxr-xr-x
		iso6937tolatin1	root	lrwxr-xr-x
		j88tosdk	root	lrwxr-xr-x
		j84tosdk	root	lrwxr-xr-x
		latin1todecmcs	root	lrwxr-xr-x
		latin1togeneralia5	root	lrwxr-xr-x
		latin1toia5	root	lrwxr-xr-x
		latin1toiso6937	root	lrwxr-xr-x
		latin1tomrtext	root	lrwxr-xr-x
		latin1tot61	root	lrwxr-xr-x
		t61togeneral	root	lrwxr-xr-x
		t61tolatin1	root	lrwxr-xr-x
		latin1toddif	root	lrwxr-xr-x
		ddiftolatin1	root	lrwxr-xr-x
		ddiftoodifq111	root	lrwxr-xr-x
		ddiftoodifq112	root	lrwxr-xr-x
		ddiftoodifq121	root	lrwxr-xr-x
		ddiftowpsplus	root	lrwxr-xr-x
		odiftoddif	root	lrwxr-xr-x

(continued on next page)

Table D–3 (Cont.) The Files Installed from a MAILbus 400 MTA Server Subset

Description	Directory	File	Ownership	Protection
		mrtexttolatin1	root	lrwxr-xr-x
		sdktoj88	root	lrwxr-xr-x
		sdktoj84	root	lrwxr-xr-x
		sdktosjis	root	lrwxr-xr-x
		sjistosdk	root	lrwxr-xr-x
		wpsplustoddif	root	lrwxr-xr-x
		wpsplustolatin1	root	lrwxr-xr-x
		w4w01t	root	lrwxr-xr-x
		w4w30f	root	lrwxr-xr-x
		w4w45f	root	lrwxr-xr-x
Softlinks to odiftoddif converter		odifq111toddif	root	lrwxr-xr-x
		odifq112toddif	root	lrwxr-xr-x
		odifq121toddif	root	lrwxr-xr-x
Options files for the odiftoddif converter[1]		q111_dap_options	root	lrw-r–r–
		q112_dap_options	root	lrw-r–r–
		q121_dap_options	root	lrw-r–r–
Warning Text File[2]	/var/mta	mta_ia5_warning_text	root	-rw-r–r–
Application services	/usr/shlib	libmtadir.so	root	lrw-r–r–

[1]These files must not be modified.

[2]This file is not a softlink.

Table D–4 The Files Installed from a MAILbus 400 MTA Base Subset

Description	Directory	File	Ownership	Protection
Shared images	/usr/shlib	libxapi.so	root	lrw-r–r–
		libxapi_remote.so	root	lrw-r–r–
Server Address	/var/mta	mta_api_server_address	root	lrw-r–r–
Reader's Comments Template	/var/mta	mta_rc_template.txt	root	lrw-r–r–
Bodypart Mapping Table	/var/mta	mta_bp_map_table.template	root	lrw-r–r–
API Release Notes	/usr/doc	mailbus400_api.release_ notes	root	lrw-r–r–

D.4 Tools Supplied with the MTA

The MTA Server subset includes the following tools:

- Accounting Decoder tool (Section D.4.1)

  This tool decodes and displays messages that are in the MTA's Accounting directory.

- Message Decoder tool (Section D.4.2)

  This tool decodes and displays messages and bodyparts that are in the MTA's bad messages directory. The Message Decoder tool can also be used to decode and display messages that are in the MTA's Archive directory.

D.4.1 Accounting Decoder Tool

The MTA stores Accounting information about messages in ASN.1 (BER) format. The Accounting Decoder tool decodes ASN.1 (BER). Use this tool to decode and display Accounting files held in the MTA's Accounting directory.

You can use this tool to display all the Accounting information for each entry in an Accounting file. Alternatively, you can use the `-b` qualifier to display the first four lines only of each entry.

Use the following command to run the Accounting Decoder tool:

```
# /usr/sbin/mta/mta_accdecoder -f /var/mta/accounting/filename [-b]
```

where ***filename*** is the name of the Accounting file.

If you move the Accounting file to some other directory, for example to prevent it being purged, then enter the new directory specification, that is, full filename including path.

D.4.2 Message Decoder Tool

The MTA stores messages on disk in ASN.1 (BER) format. The Message Decoder tool decodes the ASN.1 (BER) and displays the message. Use this tool to examine messages in the following directories:

- /var/mta/bad_msgs

  You can also use the Message Decoder tool to examine IPM bodyparts in this directory.

- /var/mta/archive
- The MTA Input and Output queues.

  /var/mta/workspace/agents/*agent_name*/oq
  /var/mta/workspace/agents/*agent_name*/iq

where *agent_name* is the name of the Agent.

You can use this tool to display a complete message. You can also use the following qualifiers:

-a	displays the header of a message in the Archive directory.
-e	decodes and displays the message envelope. Note that the envelope contains the content of the message, in its encoded form, which is also displayed.
-c	decodes and displays the content of the message.
-b	decodes and displays an IPM bodypart. Use this qualifier to display an IPM bodypart that failed conversion.
-p	parses the ASN.1 (BER) data in the file named in the command and verifies that the file contains valid ASN.1.

Use the following command to run the Message Decoder tool:

```
# /usr/sbin/mta/mta_decoder -f directory/filename [qualifier]
```

When displaying a bodypart, ***directory*** is the directory identifier, ***filename*** is the name of the bodypart file, and ***qualifier*** is `-b`.

When displaying a message, ***directory*** is the directory identifier, ***filename*** is the name of the message file, and ***qualifier*** is one of the following optional qualifiers: `-h`, `-e`, `-c`, or `-p`. If you do not specify an optional qualifier, the `-e` qualifier is used.

When displaying a message in the Archive directory, ***directory*** is the directory identifier, ***filename*** is the name of the Archive file, and ***qualifier*** is one of the following optional qualifiers: `-a`, `-h`, `-e`, `-c`, or `-p`. If you do not specify an optional qualifier, the `-e` qualifier is used.

You can also use the `-s` qualifier to display a message in the Input or Output queues for the Shared File interface. See *HP MAILbus 400 MTA Planning and Setup* for information about the Input and Output queues for the Shared File interface.

D.5 Executing NCL Scripts

In certain circumstances indicated in Part II and Part III you may have to execute the following scripts manually:

- MTA CLNS Transport Template Script:
  /var/mta/scripts/create_mta_clns_templates.ncl
- MTA CONS Transport Template Script:
  /var/mta/scripts/create_mta_cons_templates.ncl

- MTA Event Dispatching Script: /var/mta/scripts/start_mta_event_dispatching.ncl
- MTA Shutdown Script: /var/mta/scripts/stop_mta.ncl
- MTA Startup Script: /var/mta/scripts/start_mta.ncl

Use the following command to execute these scripts:

```
NCL> Do script
```

where ***script*** is the script you want to execute.

D.6 Port Number for Agents Using the API Server Over TCP/IP

Agents connecting to the MTA through the API Server over TCP/IP have a port number registered in the Tru64 UNIX services file `/etc/services`. The name registered is mta_api_server and the number of the port is 200. If this port number is used by another application, you will need to change the port number in the Tru64 UNIX services file at the MTA. Do this by editing the appropriate line in the `/etc/services` file. Make sure that you edit the file both on the systems where the Agent and the MTA are running, and on the systems where any other Agents served by the MTA are running.

D.7 Stopping and Starting the MTA

When solving MTA-related problems, or when tuning the MTA, it may be necessary to stop and start the MTA.

To stop the MTA, log into a privileged account on the node where the MTA is running and execute the MTA's shutdown script using the following command:

```
NCL> do /var/mta/scripts/stop_mta.ncl
```

To start the MTA, execute the MTA's startup script using the following command:

```
NCL> do /var/mta/scripts/start_mta.ncl
```

D.8 Restarting an MTA that is Not Responding to Management

This section describes how to restart an MTA that, for any reason, stops processing messages and fails to respond to NCL commands.

Before restarting the MTA you need to delete all of the MTA's processes. To do this, complete the following steps:

1. Log into a superuser account on the node where the MTA is running.
2. Find out the process identifiers of the MTA processes and subprocesses by entering the following command:

   ```
   # ps -e | grep mta
   ```

   This command displays the MTA processes.
3. Delete the /usr/sbin/mta/mta process of the MTA, as follows:

   ```
   # kill -9 process-id
   ```

   where ***process-id*** is the identifier of the /usr/sbin/mta/mta process.

   After a few seconds all of the MTA's processes and subprocesses will have been deleted.
4. Start the MTA again using the MTA's startup script:

   ```
   NCL> do /var/mta/scripts/start_mta.ncl
   ```

E The OpenVMS Implementation of the MTA

This appendix describes the following features that are specific to the OpenVMS VAX and OpenVMS Alpha implementations of the MAILbus 400 MTA:

- The privileges that you need in order to manage a MAILbus 400 MTA (Section E.1).
- The files that you can use (Section E.2).
- How to run the NCL scripts supplied with the MAILbus 400 MTA (Section E.3).
- The directories used by the MAILbus 400 MTA (Section E.4).
- The logical names specific to the MAILbus 400 MTA (Section E.5 and Section E.6).
- The tools supplied with the MAILbus 400 MTA (Section E.7).
- Port number for Agents using the API Server over TCP/IP, if required (Section E.8).
- How to stop and start the MAILbus 400 MTA (Section E.9).
- How to restart a MAILbus 400 MTA Server that is not responding to management (Section E.10).

This appendix also lists all the files on your system after you have installed the MAILbus 400 MTA Mgt, Server and Base components (Section E.11).

Note

In this appendix, the location and name of the directories used by the MTA is given as:

device:[MTA$*node*]

where *device* is the device that you specified when you set up the MTA and *node* is the node where you set up the MTA.

E.1 Privileges

Generally, to manage an MTA you need a privileged account, such as the SYSTEM account. In addition there are some attributes that can only be displayed, using the NCL Show command, from a privileged account. These attributes are identified in the *MTA Module Online Help* and in Appendix G.

E.2 Files You Can Use

This section lists the files that you use to set up and manage the MTA and the directory.

Table E–1 lists node specific NCL scripts and the MTA's warning text file. The files listed in Table E–1, with one exception, are copied from SYS$COMMON:[MTA.TEMPLATE_FILES] when you set up the MTA.

The file that is not copied from SYS$COMMON:[MTA.TEMPLATE_FILES] is MTA$MTS_CREATE_MTA_ENTRY.NCL. This script is created when you set up the MTA.

Table E–1 Node Specific NCL Scripts and Warning Text File

Description	File	Protection
Files in SYS$SPECIFIC:[SYS$STARTUP]		
MTA CLNS Template Script	MTA$CREATE_CLNS_TEMPLATES.NCL	RWED,RWED,RE,
MTA CONS Template Script	MTA$CREATE_CONS_TEMPLATES.NCL	RWED,RWED,RE,
MTA RFC 1006 Template Script	MTA$CREATE_RFC1006_TEMPLATES.NCL	RWED,RWED,RE,

(continued on next page)

Table E–1 (Cont.) Node Specific NCL Scripts and Warning Text File

Description	File	Protection
Files in SYS$SPECIFIC:[SYS$STARTUP]		
Example MTS Population Script	MTA$MTS_POPULATE_EXAMPLE.NCL	RWED,RWED,RE,
Populate Countries Script	MTA$MTS_POPULATE_COUNTRIES.NCL	RWED,RWED,RE,
MTA Startup Script	MTA$START.NCL	RWED,RWED,RE,
MTA Event Dispatching Script	MTA$START_EVENT_ DISPATCHING.NCL	RWED,RWED,RE,
MTA Shutdown Script	MTA$STOP.NCL	RWED,RWED,RE,
Create MTA Entry Script	MTA$MTS_CREATE_MTA_ENTRY.NCL[1]	RWED,RWED,RE,
File in *device*:[MTA$*node*]		
Warning Text File	MTA$IA5_WARNING_TEXT.TXT	RWED,RWED,RE,

[1]Generated by MTA$SERVER_SETUP.COM

Table E–2 lists the procedures you use to set up, start, and shut down the MTA.

Table E–2 MTA Setup and Startup Procedures

Description	File	Protection
Files in SYS$COMMON:[SYS$STARTUP]		
MTA Server Setup Procedure	MTA$SERVER_SETUP.COM	RWED,RWED,RE,
MTA Server Initialization Procedure	MTA$SERVER_INIT.COM	RWED,RWED,RE,
MTS Process Initialization Procedure	MTA$MTS_INIT.COM	RWED,RWED,RE,
MTA Server Shutdown Procedure	MTA$SERVER_SHUTDOWN.COM	RWED,RWED,RE,
MTA Server Startup Procedure	MTA$SERVER_STARTUP.COM	RWED,RWED,RE,

(continued on next page)

Table E–2 (Cont.) MTA Setup and Startup Procedures

Description	File	Protection
Files in SYS$COMMON:[SYS$STARTUP]		
MTA Shutdown Procedure	MTA$COMMON_SHUTDOWN.COM	RWED,RWED,RE,
MTA Startup Procedure	MTA$COMMON_STARTUP.COM	RWED,RWED,RE,
Client Startup Procedure	MTA$CLIENT_STARTUP.COM	RWED,RWED,RE,
Client CLNS Transport Template Script	MTA$CREATE_CLIENT_CLNS_ TEMPLATES.NCL	RWED,RWED,RE,

Table E–3 lists the procedures you use to run the Accounting Decoder and Message Decoder tools. See Section E.7 for the commands you need to run these tools.

Table E–3 Decoder Tool Procedures

Description	File	Protection
Files in SYS$COMMON:[SYSMGR]		
Accounting Decoder Tool Procedure	MTA$ACCDECODER.COM	RWED,RWED,RE,
Message Decoder Tool Procedure	MTA$DECODER.COM	RWED,RWED,RE,

Table E–4 shows the location of the bodypart mapping table. Instructions on how to modify the bodypart mapping table are provided in the bodypart mapping table itself.

Table E–4 Bodypart Mapping Table

Description	File	Protection
Files in SYS$COMMON:[MTA]		
Bodypart Mapping Table	MTA$BP_MAP_TABLE.TXT	RWED,RWED,RE,

E.3 Executing NCL Scripts

In certain circumstances indicated in Part II and Part III you may have to manually execute the NCL scripts listed in Table E–1.

Use the following command to execute these scripts:

```
NCL>DO script
```

where ***script*** is the NCL script you want to execute.

E.4 Directories Used by the MTA

During operation, the MTA uses several directories. You specify the location of these directories when the MTA is set up. The Accounting directory, Archive directories, and the MTA's work area directories can be located on different devices. The MTA's work area contains the bad messages, trace and workspace directories. The workspace directories are usually accessed only by the MTA.

The locations of the MTA's directories are listed in Table E–5.

Table E–5 Location of the MTA's Work Area, Accounting, and Archive Directories

Directory	Description
device:[MTA$*node*.ACCOUNTING][1]	Accounting files
device:[MTA$*node*.ARCHIVE][1]	Archived messages
device:[MTA$*node*.BAD_MSGS][1]	MPDUs or IPM bodyparts that the MTA cannot process
device:[MTA$*node*.TRACE][1]	Trace binary files
device:[MTA$*node*.WORKSPACE]	MTA's workspace (MTA use only)

[1]You can access this directory by using the appropriate logical name listed in Section E.5

E.5 Logical Names for Accounting, Archiving, Bad Message and Trace Directories

The following logical names are defined after you have set up the MTA and the MTA is operational. These logicals are for your use only and are not used by the MTA.

- MTA$ACCOUNTING

  This logical name defines the location of the MTA's Accounting directory.

- MTA$ARCHIVE

  This logical name defines the location of the MTA's Archive directories.

- MTA$BAD_MSGS

  This logical name defines the location of the MTA's bad messages directory.
- MTA$TRACE

  This logical name defines the location of the MTA's Trace directory. This directory contains the binary trace files created by the MTA when recording protocol information (see Section 16.4).

E.6 Logical Defining CLNS Address for Agents

The logical MTA_NODE defines the CLNS address of the node where the MTA is installed. Agents that use the XAPI interface need this logical name to be able to find and connect to the MTA; see Part III of *HP MAILbus 400 MTA Planning and Setup*.

E.7 Tools Supplied with the MTA

The MAILbus 400 MTA Server component includes the following tools:

- Accounting Decoder tool (Section E.7.1)

  Use this tool to decode and display messages in the MTA's Accounting directory.
- Message Decoder tool (Section E.7.2)

  Use this tool to decode and display messages in the MTA's bad messages directory and Archive directories, and in the Input and Output queues for the Shared File interface.

E.7.1 Accounting Decoder Tool

The MTA stores Accounting information about messages in ASN.1 (BER) format. The Accounting Decoder tool decodes ASN.1 (BER). Use this tool to decode and display Accounting files held in the MTA's Accounting directory.

You can use this tool to display all the Accounting information for each entry in an Accounting file. Alternatively, you can use the `-B` qualifier to display the first four lines only of each entry.

To run the Accounting Decoder tool, execute the following procedure:

```
$ @SYS$MANAGER:MTA$ACCDECODER -F MTA$ACCOUNTING:filename [qualifier]
```

where ***filename*** is the name of the Accounting file and ***qualifier*** is the optional **-B** qualifier.

E.7.2 Message Decoder Tool

The MTA stores messages on disk in ASN.1 (BER) format. The Message Decoder tool decodes the ASN.1 (BER) and displays the message. You can also use the Message Decoder tool to examine IPM bodyparts in the bad messages directory.

To decode and display a message or an IPM bodypart in the bad messages directory, execute the following procedure:

$ **@SYS$MANAGER:MTA$DECODER -F MTA$BAD_MSGS:*filename*** [***qualifier***]

where:

- ***filename*** is the name of the file containing the bad message.
- ***qualifier*** is one of the following optional qualifiers:

-E	decodes and displays the message envelope. Note that the envelope contains the content of the message, in its encoded form, which is also displayed.
-C	decodes and displays the content of the message.
-B	decodes and displays an IPM bodypart. Use this qualifier to display an IPM bodypart that failed conversion.
-P	parses the ASN.1 (BER) data in the file named in the command and verifies that the file contains valid ASN.1.

If you do not specify an optional qualifier, the **-E** qualifier is used.

You can also use the **-S** qualifier to display a message in the Input or Output queues for the Shared File interface. See Part III of *HP MAILbus 400 MTA Planning and Setup* for information about the Input and Output queues for the Shared File interface.

To decode and display a message in an Archive directory, execute the following procedure:

$ **@SYS$MANAGER:MTA$DECODER -F MTA$ARCHIVE:*filename*** [***qualifier***]

where:

- ***filename*** is the name of the Archive file.
- ***qualifier*** is one of the following optional qualifiers:

-A	displays the header of a message only.
-E	decodes and displays the message envelope. Note that the envelope contains the content of the message, in its encoded form, which is also displayed.

-C decodes and displays the content of the message.

-P parses the ASN.1 (BER) data in the file named in the command and verifies that the file contains valid ASN.1.

If you do not specify an optional qualifier, the **-E** qualifier is used.

E.8 Port Number for Agents Using the API Server Over TCP/IP

Agents connecting to the MTA through the API Server over TCP/IP have a registered name mta_api_server and a port number of 200. If you want to change the port number used by the application, you will need to change the port number as directed by the documentation relating to the TCP/IP services that you use. Make sure that you make the change on both the system where the Agent and the MTA are running, and on the systems where any other Agents served by the MTA are running.

E.9 Stopping and Starting the MTA

When solving MTA-related problems, or when tuning the MTA, it may be necessary to stop and start the MTA.

To stop the MTA, log into a privileged account on the node where the MTA is running and execute the following procedure:

```
$ @SYS$STARTUP:MTA$SERVER_SHUTDOWN
```

Note that this procedure does not delete the MTA's processes. See Section E.10 for information about deleting and creating the MTA's processes.

To start the MTA, execute the following procedure:

```
$ @SYS$STARTUP:MTA$SERVER_STARTUP
```

E.10 Restarting an MTA That is Not Responding to Management

This section describes how to restart a MAILbus 400 MTA that, for any reason, stops processing messages and fails to respond to NCL commands.

Shutdown and then restart the MTA as follows:

1. Log into a privileged account on the node where the MTA that you want to restart is running.
2. Stop the MTA's processes by executing the following procedure:

   ```
   $ @SYS$STARTUP:MTA$COMMON_SHUTDOWN
   ```

After a few seconds all of the MTA's processes are stopped. The MTA$COMMON_SHUTDOWN procedure stops the following processes:

– MTA$SERVER

– MTA$XAPI_SERVER

– MTA$MTS

3. Start the MTA again by executing the following procedure:

```
$ @SYS$STARTUP:MTA$COMMON_STARTUP
```

E.11 Files on Your System After Installation

The following tables describe the files on your system after you have installed the MAILbus 400 MTA and run the MTA setup procedure:

- Table E–6 lists the files present on your system after installation of the MAILbus 400 MTA Mgt component.
- Table E–7 lists the files present on your system after the installation of the MAILbus 400 MTA Server component.
- Table E–8 lists the files present on your system after the installation of the MAILbus 400 MTA Base component.

Note that there are other files that may be on the same node as the MTA that also have names prefixed with "MTA". In particular, the files SYS$SYSTEM:MTAAACP.EXE and SYS$LIBRARY:MTADEFS.H, which are part of the OpenVMS operating system.

Table E–6 Files on Your System After You have Installed MAILbus 400 MTA Mgt

File	Owner	Protection
Files in SYS$COMMON:[MTA]		
MTA$MTS.EXE	SYSTEM	RWED,RWED,RE,RE
MTS_START.COM	SYSTEM	RWED,RWED,RE,
Files in SYS$COMMON:[MTA.TEMPLATE_FILES]		
MTA$MTS_POPULATE_EXAMPLE.NCL	SYSTEM	RWED,RWED,RE,
MTA$MTS_POPULATE_COUNTRIES.NCL	SYSTEM	RWED,RWED,RE,

(continued on next page)

Table E–6 (Cont.) Files on Your System After You have Installed MAILbus 400 MTA Mgt

File	Owner	Protection
Files in SYS$COMMON:[SYSHLP]		
MAILBUS400_MTA.RELEASE_NOTES	SYSTEM	RWED,RWED,RE,RE
Files in SYS$COMMON:[SYS$STARTUP]		
MTA$COMMON_SHUTDOWN.COM	SYSTEM	RWED,RWED,RE,
MTA$COMMON_STARTUP.COM	SYSTEM	RWED,RWED,RE,
MTA$MTS_INIT.COM	SYSTEM	RWED,RWED,RE,

Table E–7 Files on Your System After You have Installed the MAILbus 400 MTA Server

File	Owner	Protection
Files in SYS$COMMON:[MTA]		
MTA_ACCDECODER.EXE	SYSTEM	RWED,RWED,RE,RE
MTA_DECODER.EXE	SYSTEM	RWED,RWED,RE,RE
MTA$SERVER_V6.EXE	SYSTEM	RWED,RWED,RE,RE
MTA$SERVER_V7.EXE	SYSTEM	RWED,RWED,RE,RE
MTA_START.COM	SYSTEM	RWED,RWED,RE,
MTA$XAPI_SERVER.EXE	SYSTEM	RWED,RWED,RE,RE
XAPI_SERVER_START.COM	SYSTEM	RWED,RWED,RE,
Files in SYS$COMMON:[MTA.CONVERTERS.IPM]		
DECDXTODDIF.EXE	SYSTEM	RWED,RWED,RE,RE
DECDXTOLATIN1.COM	SYSTEM	RWED,RWED,RE,RE
DECMCSTOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
DDIFTOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
DDIFTOODIFQ111.EXE	SYSTEM	RWED,RWED,RE,RE
DDIFTOODIFQ112.EXE	SYSTEM	RWED,RWED,RE,RE
DDIFTOODIFQ121.EXE	SYSTEM	RWED,RWED,RE,RE
DDIFTOWPSPLUS.EXE	SYSTEM	RWED,RWED,RE,RE
EXTERNALDEFTOBILATDEF.EXE	SYSTEM	RWED,RWED,RE,RE
EXTERNALDEFTOPOSTE.EXE	SYSTEM	RWED,RWED,RE,RE
GENERALT61TOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
GENERALTOIA5.EXE	SYSTEM	RWED,RWED,RE,RE

(continued on next page)

Table E–7 (Cont.) Files on Your System After You have Installed the MAILbus 400 MTA Server

File	Owner	Protection
Files in SYS$COMMON:[MTA.CONVERTERS.IPM]		
GENERALTOT61.EXE	SYSTEM	RWED,RWED,RE,RE
IA5TOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
ISO6937TOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TODDIF.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TODECMCS.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TOGENERALIA5.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TOIA5.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TOISO6937.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TOMRTEXT.EXE	SYSTEM	RWED,RWED,RE,RE
LATIN1TOT61.EXE	SYSTEM	RWED,RWED,RE,RE
MRTEXTTOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
ODIFQ111TODDIF.EXE	SYSTEM	RWED,RWED,RE,RE
ODIFQ112TODDIF.EXE	SYSTEM	RWED,RWED,RE,RE
ODIFQ121TODDIF.EXE	SYSTEM	RWED,RWED,RE,RE
Q111_DAP_OPTIONS.DAT	SYSTEM	RWED,RWED,RE,RE
Q112_DAP_OPTIONS.DAT	SYSTEM	RWED,RWED,RE,RE
Q121_DAP_OPTIONS.DAT	SYSTEM	RWED,RWED,RE,RE
T61TOGENERAL.EXE	SYSTEM	RWED,RWED,RE,RE
T61TOLATIN1.EXE	SYSTEM	RWED,RWED,RE,RE
W4W01T.EXE	SYSTEM	RWED,RWED,RE,RE
W4W30F.EXE	SYSTEM	RWED,RWED,RE,RE
W4W45F.EXE	SYSTEM	RWED,RWED,RE,RE
WPSPLUSTODDIF.EXE	SYSTEM	RWED,RWED,RE,RE
WPSPLUSTOLATIN1.COM	SYSTEM	RWED,RWED,RE,RE
Files in SYS$COMMON:[MTA.TEMPLATE_FILES]		
MTA$CREATE_CLNS_TEMPLATES.NCL	SYSTEM	RWED,RWED,RE,
MTA$CREATE_CONS_TEMPLATES.NCL	SYSTEM	RWED,RWED,RE,
MTA$CREATE_RFC1006_TEMPLATES.NCL	SYSTEM	RWED,RWED,RE,
MTA$CREATE_EXTDEF_BODYPARTS.NCL	SYSTEM	RWED,RWED,RE,
MTA$IA5_WARNING_TEXT.TXT	SYSTEM	RWED,RWED,RE,
MTA$START.NCL	SYSTEM	RWED,RWED,RE,
MTA$START_EVENT_DISPATCHING.NCL	SYSTEM	RWED,RWED,RE,
MTA$STOP.NCL	SYSTEM	RWED,RWED,RE,

(continued on next page)

Table E–7 (Cont.) Files on Your System After You have Installed the MAILbus 400 MTA Server

File	Owner	Protection
Files in SYS$COMMON:[SYSLIB]		
MTA$DIR_CMA_SHR.EXE	SYSTEM	RWED,RWED,RWED,RE
MTA$DIR_SHR.EXE	SYSTEM	RWED,RWED,RWED,RE
Files in SYS$COMMON:[SYSMGR]		
MTA$ACCDECODER.COM	SYSTEM	RWED,RWED,RE,
MTA$DECODER.COM	SYSTEM	RWED,RWED,RE,
Files in SYS$COMMON:[SYS$STARTUP]		
MTA$SERVER_INIT.COM	SYSTEM	RWED,RWED,RE,
MTA$SERVER_SETUP.COM	SYSTEM	RWED,RWED,RE,
MTA$SERVER_SHUTDOWN.COM	SYSTEM	RWED,RWED,RE,
MTA$SERVER_STARTUP.COM	SYSTEM	RWED,RWED,RE,
Files in SYS$COMMON:[SYSTEST]		
MTA$VP.COM	SYSTEM	RWED,RWED,RE,
MTA$MTAMAIL.EXE	SYSTEM	RWED,RWED,RE,

Table E–8 Files on Your System After You have Installed MAILbus 400 MTA Base

File	Owner	Protection
Files in SYS$COMMON:[SYSLIB]		
MTA$XAPI_REM_SHR.EXE	SYSTEM	RWED,RWED,RWED,RE
MTA$XAPI_SHR.EXE	SYSTEM	RWED,RWED,RWED,RE
MTA$XAPI_CMA_SHR.EXE	SYSTEM	RWED,RWED,RWED,RE
Files in SYS$COMMON:[SYS$STARTUP]		
MTA$CLIENT_STARTUP.COM	SYSTEM	RWED,RWED,RE,
MTA$CREATE_CLIENT_CLNS_ TEMPLATES.NCL	SYSTEM	RWED,RWED,RE,

(continued on next page)

Table E–8 (Cont.) Files on Your System After You have Installed MAILbus 400 MTA Base

File	Owner	Protection
Files in SYS$COMMON:[SYSHLP]		
MAILBUS400_API.RELEASE_NOTES	SYSTEM	RWED,RWED,RE,RE
MTA$RC_TEMPLATE.TXT	SYSTEM	RWED,RWED,RE,RE
Files in SYS$COMMON:[MTA]		
MTA$BP_MAP_TABLE.TEMPLATE	SYSTEM	RWED,RWED,RE,

F
Routing Examples

This appendix gives examples of how the MAILbus 400 MTA uses routing information contained in the directory and in Agent and Peer MTA entities. For routing to work, the relevant routing information must be present in the directory and in the appropriate Agent and Peer MTA entities. If the MAILbus 400 MTA cannot find this information, it cannot route the message, and so generates a non-delivery report.

For the purposes of these examples, it is assumed that area routing is not implemented. For a description of area routing, see Part II of *HP MAILbus 400 MTA Planning and Setup*.

These examples show the information that the MTA uses at different stages when routing a message to a recipient. Note that this appendix describes the behavior of the MTA for these specific examples and that the MTA can behave differently in other situations.

F.1 Routing a Message Within a Routing Domain

In this example, a user in the ACME routing domain addresses a message to another user in the same routing domain.

The recipient's O/R address on the message is as follows:

```
Recipient = "C=NZ;A=NZ-PTT;P=ACME;O=ACME;OU1=WELL;CN=KIM YIP"
```

The MAILbus 400 MTA routes the message to this recipient as follows:

1. The MTA looks in the directory for an O/R address entry that matches the recipient's O/R address on the message. The O/R address entry contains a Routing Instruction that identifies the recipient's MTA, as follows:

```
NAME = "C=NZ;A=NZ-PTT;P=ACME;O=ACME;OU1=WELL;CN=KIM YIP"
ROUTING INSTRUCTION [ACTION = DELIVER,
                     SERVER MTA = "WELL.NODE7",
                     AGENT = "MYAGENT",
                     DEFINITIVE ORADDRESS = ""]
TYPE = USER
```

The MTA checks whether the MTA named in the Routing Instruction (WELL.NODE7) is its own name. If it is, the MTA finds the name of the User Agent named in the Routing Instruction. To deliver the message, the MTA must have an Agent entity that corresponds to this User Agent. In the example, this Agent entity is:

```
AGENT "MYAGENT"
```

The Agent entity contains information about the User Agent, such as the password used by the User Agent. The MTA needs this information to contact the User Agent and deliver the message.

2. If the MTA named in the Routing Instruction is different from the MTA that is currently responsible for the message (referred to here as the routing MTA), the routing MTA looks in the directory for an MTA entry that matches the MTA named in the Routing Instruction.

   The routing MTA checks the MTA entry to see whether it describes a single MTA or an MTA set. If it is a single MTA, the routing MTA uses the Presentation address and password specified in the MTA entry to transfer the message to the specified MTA.

3. If the MTA named in the Routing Instruction is an MTA set, and the routing MTA is a member of the set, the routing MTA uses the User Agent named in the Routing Instruction of the recipient's O/R address, and the information in the corresponding Agent entity, to deliver the message.

4. If the MTA named in the Routing Instruction is an MTA set, but the routing MTA is not a member of that set, the routing MTA selects a member of the MTA set to which it attempts to transfer the message. It then looks in the directory for an entry for the selected MTA and uses the Presentation address and password in that entry to transfer the message to that MTA.

5. When the message arrives at the MTA selected in step 4, the new MTA becomes the routing MTA. The new routing MTA then completes steps 1 to 4 again until the message arrives at the destination MTA, that is, the MTA that can deliver the message.

F.2 Routing a Message to Another Routing Domain

In this example, a user in the ACME routing domain addresses a message to a user in the MACE routing domain. The recipient's O/R address on the message is:

```
Recipient = "C=NZ;A=NZ-PTT;P=MACE;O=MACE;CN=HEIDI BRAUN"
```

The MAILbus 400 MTA routes the message to this recipient as follows:

1. The routing MTA looks in the directory for an O/R address entry that matches the recipient's O/R address on the message. In this case, it cannot find an exact match, so it discounts the last term from the O/R address and looks for an entry that matches the partial O/R address. It continues to do this until it finds a matching entry in the directory, with a Routing Instruction as follows:

   ```
   NAME = "C=NZ;A=NZ-PTT;P=MACE"
   ROUTING INSTRUCTION [ACTION = TRANSFER TO DOMAIN,
                        SERVER DOMAIN = "MACE"]
   TYPE = USER
   ```

   The matching O/R address entry contains a Routing Instruction that identifies a Domain entry in the directory. The Domain entry should contain a Routing Instruction that identifies the boundary MTA that can send the message to the other routing domain, as follows:

   ```
   NAME = "MACE"
   ROUTING INSTRUCTION [ACTION = TRANSFER TO DOMAIN,
                        BOUNDARY MTA = "WELL.MTA-NODE6"]
   ```

   The routing MTA checks whether the name of the boundary MTA is its own name. If it is, it attempts to transfer the message to a peer MTA in the MACE routing domain. To do this, it must have a Peer MTA entity that represents a peer MTA in the MACE routing domain. To find the correct Peer MTA entity, the boundary MTA looks through the attributes of its Peer MTA entities until it finds one that contains a Peer Domain attribute that matches the name of the Domain entry, as follows:

   ```
   PEER DOMAIN = "MACE"
   ```

   The Peer MTA entity contains information about the peer MTA in the other routing domain, such as the password used by the peer MTA and the application context that it uses. The MTA needs the information in the Peer MTA entity to contact the peer MTA and transfer the message to it.

2. If the boundary MTA is different from the routing MTA, the routing MTA looks in the directory for an MTA entry for the boundary MTA. It then checks whether the boundary MTA is an MTA set, as described in step 2 of Section F.1.

   The routing MTA then uses the Presentation address and the password named in the MTA entry for the boundary MTA (or for the MTA set member selected by the routing MTA) to transfer the message to that MTA. This MTA, in turn, uses the information in its Peer MTA entity to contact the peer MTA, as described in step 1 of this section.

F.3 Routing a Message to a Postal Routing Domain

In this example, a user in the ACME routing domain addresses a message to a user in the postal routing domain ISLAND-POST. The recipient's O/R address on the message is:

```
Recipient = "C=NZ;A=NZ-PTT;PD-C=NZ;PD-SN=ISLAND-POST;PD-A1=Fred Zimmer; -
            PD-A2=92 Wood Way;PD-A3=Foxton;PD-A4=New Zealand;PD-PC=NZ9236"
```

The MAILbus 400 MTA routes the message to this recipient as follows:

1. The routing MTA looks in the directory for an O/R address entry that matches the recipient's O/R address on the message. In this case, it cannot find an exact match, so it discounts the last term from the O/R address and looks for an entry that matches the partial O/R address. It continues to do this until it finds a matching entry in the directory, with Routing Instructions as follows:

   ```
   NAME = "C=NZ;A=NZ-PTT"
   POSTAL ROUTING INSTRUCTION [ACTION = TRANSFER TO DOMAIN,
                               SERVER DOMAIN = "ISLAND-POST"]
   ROUTING INSTRUCTION [ACTION = TRANSFER TO DOMAIN,
                        SERVER DOMAIN = "NZ-PTT"]
   TYPE = USER
   ```

   In this case a routing instruction which is specific to the postal O/R address form is present. The MTA uses this instruction in preference to the routing instruction specified generally for the other O/R address forms because the recipient has a postal O/R address.

   The Domain entry specified as server domain (ISLAND-POST) should contain a Routing Instruction that identifies the boundary MTA that can send the message to the ISLAND-POST routing domain, as follows:

   ```
   NAME = "ISLAND-POST"
   ROUTING INSTRUCTION [ACTION = TRANSFER TO DOMAIN,
                        BOUNDARY MTA = "WELL.MTA-NODE8"]
   ```

   The routing MTA checks whether the name of the boundary MTA is its own name. If it is, it attempts to transfer the message to a peer MTA in the ISLAND-POST routing domain. To do this, it must have a Peer MTA entity that represents a peer MTA in the ISLAND-POST routing domain. To find the correct Peer MTA entity, the boundary MTA looks through the attributes of its Peer MTA entities until it finds one that contains a Peer Domain attribute that matches the name of the Domain entry, as follows:

   ```
   PEER DOMAIN = "ISLAND-POST"
   ```

The Peer MTA entity contains information about the peer MTA in the ISLAND-POST routing domain that can reach the physical delivery access unit (PDAU) for the ISLAND-POST delivery service. The Peer MTA entity contains information such as the password used by the peer MTA and the application context that it uses. The MTA needs the information in the Peer MTA entity to contact the peer MTA and transfer the message to it.

2. If the boundary MTA is different from the routing MTA, the routing MTA looks in the directory for an MTA entry for the boundary MTA. It then checks whether the boundary MTA is an MTA set, as described in step 2 of Section F.1.

   The routing MTA then uses the Presentation address and the password named in the MTA entry for the boundary MTA (or for the MTA set member selected by the routing MTA) to transfer the message to that MTA. This MTA, in turn, uses the information in its Peer MTA entity to contact the peer MTA, as described in step 1 of this section.

G
MTA Module Entities and Attributes

This appendix lists the characteristic, status, and identifier attributes of each entity in the MTA module. The tables in this appendix list the characteristic, status, and identifier attributes of the following entities:

- MTA entity (Table G–1)
- Agent entity (Table G–2)
- Bodypart entity (Table G–3)
- Converter entity (Table G–4)
- Deferred Message entity (Table G–5)
- Processed Message entity (Table G–6)
- MPDU entity (Table G–7)
- Peer MTA entity (Table G–8)
- Activity entity (Table G–9)

Counter attributes and events are listed in the following tables:

- Table 14–3

  Lists the counters that record statistical information about MPDUs and associations.
- Table 14–2

  Lists each event and its related counter.

Each table in this appendix lists each attribute and gives a summary of the NCL commands and the datatype or value. See the *MTA Module Online Help* for a full description of all attributes and their value syntaxes.

Table G–1 MTA Entity Attributes

Attribute	Show	Set	Datatype or Value
MTS	√	-	Printable String
Name	√	-	Printable String
Presentation Address	√[1]	-	[2]
Session Address	√[1]	-	[2]
Accounting State	√	√	On, Off
Accounting Purge Interval	√	√	Binary Relative Time
Maximum Agent Connections	√	√	Integer
Contact Name	√	√	Latin1
Template Name	√	√	See the *MTA Module Online Help*
Transport Service Options	√	√	See the *MTA Module Online Help*
Delivery Accounting Filter	√	√[3]	See the *MTA Module Online Help*
Export Accounting Filter	√	√[3]	See the *MTA Module Online Help*
Import Accounting Filter	√	√[3]	See the *MTA Module Online Help*
Submission Accounting Filter	√	√[3]	See the *MTA Module Online Help*
Message History State	√	√	On, Off
Message History Purge Interval	√	√	Binary Relative Time
Initial Transfer Retry Interval	√	√	Binary Relative Time
Maximum Automatically Configured Peer MTAs	√	√	Integer
Maximum Inbound Transfer Associations	√	√	Integer
Maximum Idle Inbound Transfer Association Interval	√	√	Binary Relative Time

[1]Not without privileges

[2]See the DECnet/OSI network documentation

[3]You can also use the Add and Remove Commands

(continued on next page)

Table G–1 (Cont.) MTA Entity Attributes

Attribute	Show	Set	Datatype or Value
Maximum Outbound Parallel Transfer Associations	√	√	Integer
Maximum Outbound Transfer Associations	√	√	Integer
Maximum Idle Outbound Transfer Associations Interval	√	√	Binary Relative Time
Maximum Transfer Associations	√	√	Integer
Maximum Transfer Lookahead	√	√	Integer
Maximum Transfer Retry Interval	√	√	Binary Relative Time
Local MPDU Expiry Interval	√	√	Binary Relative Time
Nonurgent MPDU Expiry Interval	√	√	Binary Relative Time
Normal MPDU Expiry Interval	√	√	Binary Relative Time
Maximum Message Processors	√	√	Integer
Version	√	-	Version Number
State	√	-	On, Off, Enabling, Disabling
UID	√	-	Unique identifier

Table G–2 Agent Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	See the *MTA Module Online Help*
Type	√	-	XAPI, Shared File 1984, Shared File 1992
Archive	√	√	Off, Inbound, Outbound, Inbound and Outbound
Invocation Filename	√[1]	√	File name
Password	-	√	Printable String

[1]Not without privileges

(continued on next page)

Table G–2 (Cont.) Agent Entity Attributes

Attribute	Show	Set	Datatype or Value
State	√	-	On, Off
UID	√	-	Unique identifier

Table G–3 Bodypart Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	Alphanumeric
Encoded Information Types	√	-	See the *MTA Module Online Help*
Identifier	√	-	See the *MTA Module Online Help*

Table G–4 Converter Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	Alphanumeric
Lossy	√	-	True, False
Source	√	-	Alphanumeric
Steps	√	-	Alphanumeric
Target	√	-	Alphanumeric

Table G–5 Deferred Message Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	Message identifier
Deferred Until	√	-	Binary Absolute Time
Originator	√[1]	-	O/R address
Priority	√	-	Urgent, Nonurgent Normal,
Recipients	√[1]	-	Set of O/R addresses
Size	√	-	Integer

[1]Not without privileges

(continued on next page)

Table G–5 (Cont.) Deferred Message Entity Attributes

Attribute	Show	Set	Datatype or Value
Submission Time	√	-	Binary Absolute Time

Table G–6 Processed Message Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	Message identifier
Recipient Information	√[1]	-	See the *MTA Module Online Help*

[1]Not without privileges

Table G–7 MPDU Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	See the *MTA Module Online Help*
Arrival Time	√	-	Binary Absolute Time
Expiry Time	√	-	Binary Absolute Time
Message Identifier	√	-	Message identifier
Originator	√[1]	-	O/R address
Priority	√	-	Urgent, Nonurgent, Normal
Recipients	√[1]	-	Set of O/R addresses
Retry Time	√	-	Binary Relative Time
Size	√	-	Integer
State	√	-	See the *MTA Module Online Help*
Target	√	-	See the *MTA Module Online Help*
Type	√	-	Message, Probe, Report

[1]Not without privileges

Table G–8 Peer MTA Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	See the *MTA Module Online Help*
Type	√	-	Automatically Configured, Manually Configured
Application Context[1]	√	√	See the *MTA Module Online Help*
Archive[1]	√	√	Off, Inbound, Outbound, Inbound and Outbound
Contact Name[1]	√	√	Latin1
Direction[1]	√	√	Inbound, Outbound, Inbound and Outbound
Peer Domain[1]	√	√	Printable String
Presentation Address[1]	√[2]	√	[3]
Session Address[1]	√[2]	√	[3]
Local Name[1]	√	√	IA5 Graphic Subset
Local Password[1]	-	√	IA5 Graphic Subset or Octet
Peer Name[1]	√	√	IA5 Graphic Subset
Peer Password[1]	-	√	IA5 Graphic Subset or Octet
Transport Service Options	√	√	See the *MTA Module Online Help*
Template Name[1]	√	√	See the *MTA Module Online Help*
Maximum Inbound Parallel Transfer Associations[1]	√	√	Integer
Maximum Outbound Parallel Transfer Associations[1]	√	√	Integer
Trace[1]	√	√	On, Off

[1]This attribute is only applicable to Peer MTA entities that are manually created

[2]Not without privileges

[3]See the DECnet/OSI network documentation

(continued on next page)

Table G–8 (Cont.) Peer MTA Entity Attributes

Attribute	Show	Set	Datatype or Value
Transfer In Accounting Filter[1]	√	√[4]	See the *MTA Module Online Help*
Transfer Out Accounting Filter [1]	√	√[4]	See the *MTA Module Online Help*
Retry Count	√	-	Integer
Retry Time	√	-	Binary Absolute Time
State	√	-	On, Off
UID	√	-	Unique identifier

[1]This attribute is only applicable to Peer MTA entities that are manually created

[4]You can also use the Add and Remove commands

Table G–9 Activity Entity Attributes

Attribute	Show	Set	Datatype or Value
Name	√	-	See the *MTA Module Online Help*
Application Context	√	-	See the *MTA Module Online Help*
Creation Time	√	-	Binary Absolute Time
Current MPDU	√	-	MPDU identifier
Direction	√	-	Inbound, Outbound, Inbound and Outbound
Interruption Reason	√	-	Text
Port	√	-	See the *MTA Module Online Help*
SCID	√	-	Unique identifier
State	√	-	Idle, Active, Interrupted
UID	√	-	Unique identifier

H
Characters in Character Sets

This appendix shows the characters that comprise the different character sets used to specify the value of attributes of entities of the MTA and MTS modules.

H.1 Printable String

Table H–1 shows the characters that can appear in the printable string character set, which is specified in CCITT Recommendation X.208.

Table H–1 Printable String Characters

Name	Character
Capital letters	A, B, . . . Z
Small letters	a, b, . . . z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Space	(space)
Apostrophe	’
Left Parenthesis	(
Right Parenthesis	)
Plus Sign	+
Comma	,
Hyphen	-
Full Stop (period)	.
Solidus	/
Colon	:
Equal Sign	=

(continued on next page)

Table H–1 (Cont.) Printable String Characters

Name	Character
Question Mark	?

H.2 IA5 Graphic Subset

Table H–2 shows the characters that comprise the IA5 graphic subset, which is specified in CCITT Recommendation T.50.

Table H–2 IA5 Graphic Subset Characters

Name	Character
Capital letters	A, B, . . . Z
Small letters	a, b, . . . z
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Exclamation mark	!
Quotation mark	"
Number sign	#
Currency sign	¤
Percent sign	%
Ampersand	&
Apostrophe	’
Left parenthesis	(
Right parenthesis	)
Asterisk	*
Plus sign	+
Comma	,
Hyphen	-
Full stop (period)	.
Solidus	/
Colon	:
Semi-colon	;

(continued on next page)

Table H–2 (Cont.) IA5 Graphic Subset Characters

Name	Character
Less than sign	<
Equals sign	=
Greater than sign	>
Question mark	?
Commercial at	@
Left square bracket	[
Reverse solidus	\
Right square bracket	]
Upward arrow head	^
Underline	_
Grave accent	(for example, with small e) è
Left curly bracket (brace)	{
Vertical line	|
Right curly bracket (brace)	}
Overline	‾
Space	(space)

H.3 Teletex String

The teletex character set, specified in CCITT Recommendation T.61, consists of graphic and control characters. The graphic characters include not only decimal digits and basic Latin letters, but also accented letters and alphabetic characters from non-Latin alphabets. Table H–3 shows the graphic characters from the teletex character set. For technical reasons, some of these are not shown, and the table shows their name only.

The hexadecimal values shown in Table H–3 describe the position of each character in the 16x16 code table for graphic characters shown in CCITT Recommendation T.61. The value for each character consists of the column number (0 through F) and the row number (0 through F). Thus, for example, capital A appears in column 4, row 1, and capital B in column 4, row 2. Capital P appears in column 5, row 0, and capital Z in column 5, row A.

Table H–3 Teletex String Graphic Characters

Name	Character	Hexadecimal Value
Capital Latin letters	A, B, . . . Z	41 to 5A
Small Latin letters	a, b, . . . z	61 to 7A
Capital Æ diphthong	Æ	E1
Small æ diphthong	æ	F1
Capital D with stroke	Đ	E2
Small d with stroke	(not shown)	F2
Small eth (Icelandic)	ð	F3
Capital H with stroke	(not shown)	E4
Small h with stroke	(not shown)	F4
Small i without dot	(not shown)	F5
Capital IJ ligature	(not shown)	E6
Small ij ligature	(not shown)	F6
Capital L with middle dot	(not shown)	E7
Small l with middle dot	(not shown)	F7
Capital L with stroke	(not shown)	E8
Small l with stroke	(not shown)	F8
Capital O with slash	Ø	E9
Small o with slash	ø	F9
Capital Œ ligature	Œ	EA
Small œ ligature	œ	FA
Small sharp s (German)	ß	FB
Capital thorn (Icelandic)	Þ	EC
Small thorn (Icelandic)	þ	FC
Capital T with stroke	(not shown)	ED
Small t with stroke	(not shown)	FD
Capital eng (Lapp)	(not shown)	EE
Small eng (Lapp)	(not shown)	FE
Small n with apostrophe	’n	EF
Small k (Greenlandic)	(not shown)	F0

(continued on next page)

Table H–3 (Cont.) Teletex String Graphic Characters

Name	Character	Hexadecimal Value
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9	30 to 39
Currency symbol	¤	A8
Pound sign	£	A3
Dollar sign	$	A4
Cent sign	¢	A2
Yen sign	¥	A5
Space	(space)	20
Exclamation mark	!	21
Inverted exclamation mark	¡	A1
Quotation mark	"	22
Apostrophe	’	27
Left parenthesis	(	28
Right parenthesis	)	29
Comma	,	2C
Low line	_	5F
Hyphen or minus sign	-	2D
Full stop (period)	.	2E
Solidus	/	2F
Colon	:	3A
Semicolon	;	3B
Question mark	?	3F
Inverted question mark left	¿	BF
Angle quotation mark left	«	AB
Angle quotation mark right	»	BB
Plus sign	+	2B
Plus/minus sign	±	B1
Less-than sign	<	3C
Equals sign	=	3D
Greater-than sign	>	3E

(continued on next page)

Table H–3 (Cont.) Teletex String Graphic Characters

Name	Character	Hexadecimal Value
Divide sign	÷	B8
Multiply sign	×	B4
Superscript 2	^{2}	B2
Superscript 3	^{3}	B3
Fraction one half	½	BD
Fraction one quarter	¼	BC
Fraction three quarters	¾	BE
Number sign	#	A6
Percent sign	%	25
Ampersand	&	26
Asterisk	*	2A
Commercial at	@	40
Left square bracket	[	5B
Right square bracket	]	5D
Vertical line	|	7C
Micro sign	μ	B5
Ohm sign	Ω	E0
Degree sign	°	B0
Masculine ordinal indicator	º	EB
Feminine ordinal indicator	ª	E3
Section sign	§	A7
Paragraph sign (pilcrow)	¶	B6
Middle dot	·	B7

You can use the hexadecimal value to specify any teletex character. Normally, you would only use the hexadecimal value to specify a teletex character that is not part of the ISO Latin 1 character set. When you specify a character as a hexadecimal value, enclose the value for each character in backslash characters (\). For example, to specify the feminine ordinal indicator, enter `\E3\`.

The graphic representation of an accented letter consists of the combination of the basic Latin letter and a diacritical (non-spacing) mark. Table H–4 shows the accents that are included in the teletex character set.

Table H–4 Accents in the Teletex Character Set

Accent	Hexadecimal Value
Acute	C2
Grave	C1
Circumflex	C3
Diaeresis or umlaut	C8
Tilde	C4
Caron	CF
Breve	C6
Double acute accent	CD
Ring	CA
Dot	C7
Macron	C5
Cedilla	CB
Non-spacing underline	CC
Ogonek	CE

To specify a teletex character that includes a diacritical mark and that is not part of the ISO Latin 1 character set, specify the hexadecimal value of the diacritical mark, as shown in Table H–4, then that of the basic character. For example, to specify the letter y with a circumflex, enter `\C379\`.

H.4 Numeric String

Table H–5 shows the characters that comprise the numeric string character set, which is specified in CCITT Recommendation X.208.

Table H–5 Numeric String Characters

Name	Character
Digits	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Space	(space)

H.5 Octet String

An octet string consists of one or more octets. An octet is described by two hexadecimal digits (0-9, A-F).

When using an octet string to enter teletex characters, enclose the hexadecimal representation of each teletex character in backslash characters (\). For example, to specify the capital H with stroke and the small ij ligature enter: `\E4\\F6\`.

Index