

HP NonStop Pathway/iTS System Management Manual

Abstract

This manual describes the management interface to the HP NonStop™ Pathway/iTS product; it is intended for system managers and operators. It provides instructions and guidelines for configuring and controlling the Pathway/iTS objects (those related to terminal control processes (TCPs), terminals, and intelligent devices) and for monitoring the status and performance of those objects. It also provides the syntax and complete descriptions of all PATHCOM commands for Pathway/iTS objects and cause, effect, and recovery information for all TCP error messages.

Product Version

Pathway/iTS 1.1

Supported Release Version Updates (RVUs)

This publication supports J06.03 and all subsequent J-series RVUs and H06.03 and all subsequent H-series RVUs, until otherwise indicated by its replacement publications.

Part Number	Published
426748-005	May 2009

Document History

Part Number	Product Version	Published
426748-001	Pathway/iTS 1.0	October 2000
426748-002	Pathway/iTS 1.0	February 2006
426748-003	Pathway/iTS 1.1	May 2007
426748-004	Pathway/iTS 1.1	May 2008
426748-005	Pathway/iTS 1.1	May 2009

Legal Notices

© Copyright 2009 Hewlett-Packard Development Company L.P.

Confidential computer software. Valid license from HP required for possession, use or copying. Consistent with FAR 12.211 and 12.212, Commercial Computer Software, Computer Software Documentation, and Technical Data for Commercial Items are licensed to the U.S. Government under vendor's standard commercial license.

The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Export of the information contained in this publication may require authorization from the U.S. Department of Commerce.

Microsoft, Windows, and Windows NT are U.S. registered trademarks of Microsoft Corporation.

Intel, Itanium, Pentium, and Celeron are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Java is a U.S. trademark of Sun Microsystems, Inc.

Motif, OSF/1, UNIX, X/Open, and the "X" device are registered trademarks and IT DialTone and The Open Group are trademarks of The Open Group in the U.S. and other countries.

Open Software Foundation, OSF, the OSF logo, OSF/1, OSF/Motif, and Motif are trademarks of the Open Software Foundation, Inc.

OSF MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE OSF MATERIAL PROVIDED HEREIN, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

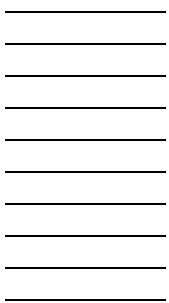
OSF shall not be liable for errors contained herein or for incidental consequential damages in connection with the furnishing, performance, or use of this material.

© 1990, 1991, 1992, 1993 Open Software Foundation, Inc. This documentation and the software to which it relates are derived in part from materials supplied by the following:

© 1987, 1988, 1989 Carnegie-Mellon University. © 1989, 1990, 1991 Digital Equipment Corporation. © 1985, 1988, 1989, 1990 Encore Computer Corporation. © 1988 Free Software Foundation, Inc. © 1987, 1988, 1989, 1990, 1991 Hewlett-Packard Company. © 1985, 1987, 1988, 1989, 1990, 1991, 1992 International Business Machines Corporation. © 1988, 1989 Massachusetts Institute of Technology. © 1988, 1989, 1990 Mentat Inc. © 1988 Microsoft Corporation. © 1987, 1988, 1989, 1990, 1991, 1992 SecureWare, Inc. © 1990, 1991 Siemens Nixdorf Informationssysteme AG. © 1986, 1989, 1996, 1997 Sun Microsystems, Inc. © 1989, 1990, 1991 Transarc Corporation.

This software and documentation are based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California. OSF acknowledges the following individuals and institutions for their role in its development: Kenneth C.R.C. Arnold, Gregory S. Couch, Conrad C. Huang, Ed James, Symmetric Computer Systems, Robert Elz. © 1980, 1981, 1982, 1983, 1985, 1986, 1987, 1988, 1989 Regents of the University of California.

Printed in the US



HP NonStop Pathway/iTS System Management Manual

Index	Examples	Figures	Tables
-----------------------	--------------------------	-------------------------	------------------------

[Legal Notices](#)

[What's New in This Manual](#) xi

[Manual Information](#) xi

[New and Changed Information](#) xi

[About This Manual](#) xv

[Who Should Read This Manual](#) xv

[Related Documentation](#) xv

[Notation Conventions](#) xvi

[1. Introduction to Pathway/iTS System Management](#)

[Supported Release Version Updates \(RVUs\)](#) 1-1

[Which Sections Do You Need?](#) 1-2

[Pathway Environment Overview](#) 1-4

[Objects and Processes Provided by TS/MP](#) 1-4

[Objects and Processes Provided by Pathway/iTS](#) 1-5

[Pathway Environment Configurations](#) 1-6

[Distributing a Pathway Environment](#) 1-8

[TS/MP Objects and Processes](#) 1-10

[PATHMON Object](#) 1-10

[PATHCOM Processes](#) 1-11

[SERVER Objects](#) 1-11

[LINKMON Processes](#) 1-11

[Pathway/iTS Objects and Processes](#) 1-12

[TCP Objects](#) 1-12

[TERM and PROGRAM Objects](#) 1-12

[Router Processes](#) 1-13

[Transaction Sources](#) 1-13

[Browser-Based Web Clients](#) 1-14

[Personal Computers and Workstations](#) 1-15

[External TCPs](#) 1-15

[Intelligent Devices](#) 1-17

1. Introduction to Pathway/iTS System Management (continued)[SNA Devices](#) 1-17[Unsupported or Special-Function I/O Devices](#) 1-17[The LINKMON Process](#) 1-18[Requester-Server Communication](#) 1-18[System Management Tasks](#) 1-19**2. Configuring Pathway/iTS Objects**[Configuration Overview](#) 2-1[Configuring Limits for Pathway/iTS Objects](#) 2-4[Specifying Limits](#) 2-4[Specifying Node Independence](#) 2-5[Configuring TCP, TERM, and PROGRAM Objects](#) 2-6[Configuring TCPs](#) 2-6[Defining Attributes](#) 2-7[Distributing the Transaction Load Across TCPs](#) 2-9[Requesting Error Dumping](#) 2-9[Configuring TCPs for a Customized TCP Object Library](#) 2-10[Configuring TERM and PROGRAM Objects](#) 2-10[Configuring TERM Objects](#) 2-11[Configuring PROGRAM Objects](#) 2-12[Communication Between PATHMON Environments](#) 2-14**3. Starting and Stopping Pathway/iTS Objects**[Starting Router Processes](#) 3-1[Starting PATHMON-Controlled Objects](#) 3-2[Starting TCPs](#) 3-4[Starting TERM Objects](#) 3-4[Starting Multiple TCPs and TERM Objects in Parallel](#) 3-5[Starting PROGRAM Objects](#) 3-5[Stopping Router Processes](#) 3-7[Stopping PATHMON-Controlled Objects](#) 3-7[Stopping TCPs](#) 3-8[Stopping TERM Objects](#) 3-8[Stopping Multiple TCPs and TERM Objects in Parallel](#) 3-9[Pathway/iTS Objects and the SHUTDOWN2 Command](#) 3-10[Specifying the ORDERLY Option](#) 3-11[Specifying the ABORT Option](#) 3-12[Specifying the IMMEDIATE Option](#) 3-12

4. Maintaining Pathway/iTS Objects

<u>System Maintenance Tasks</u>	4-1
<u>Displaying Information About Pathway/iTS Objects</u>	4-1
<u>A System Management Scenario</u>	4-2
<u>Displaying Configuration Information</u>	4-3
<u>Displaying Status Information</u>	4-6
<u>Displaying Statistics Information</u>	4-10
<u>Reconfiguring Pathway/iTS Objects</u>	4-13
<u>Specifying New Limits</u>	4-14
<u>Adding, Altering, and Deleting Objects</u>	4-14
<u>Changing Backup CPUs and Dump Files</u>	4-15
<u>Exchanging Primary and Backup CPUs</u>	4-16
<u>Logging Status and Error Information</u>	4-16
<u>Managing Exception Conditions</u>	4-17
<u>Using PATHCOM Commands</u>	4-17
<u>Using EMS Event Messages</u>	4-18
<u>Managing Links</u>	4-19
<u>Understanding the Causes of Link Dissolution</u>	4-19
<u>Improving Performance</u>	4-20
<u>Use External TCPs to Manage Terminals</u>	4-20
<u>Prevent TCLPROG File Checking</u>	4-21
<u>Improving Ready-Time-to-Busy-Time Ratio</u>	4-21
<u>Information to Include When Reporting Problems</u>	4-21
<u>TCP-Specific Problems</u>	4-21
<u>Terminal-Specific Problems</u>	4-23
<u>SCREEN-COBOL-Specific Problems</u>	4-23
<u>SCUP-Specific Problems</u>	4-24
<u>Keeping Development and Production Separate</u>	4-24
<u>Sending Messages to Users</u>	4-24
<u>Migrating Pathway/iTS Objects to a Different System</u>	4-25

5. Tuning Your System Using Statistics

<u>Statistics and System Tuning</u>	5-1
<u>TCP Tasks</u>	5-1
<u>Link Management</u>	5-2
<u>Memory Management and Allocation</u>	5-4
<u>Checkpointing</u>	5-6
<u>Gathering Statistics</u>	5-6
<u>TCP Statistics</u>	5-7

5. Tuning Your System Using Statistics (continued)

POOL INFO	5-8
AREA INFO	5-12
QUEUE INFO	5-14
REQ CNT	5-16
% WAIT	5-16
Terminal Statistics	5-17
I/O Info	5-18
Area Info	5-23
Response Time Info	5-24
Frequency Distribution	5-25

6. Examples of System Management Tasks

Example Task Overview	6-1
Configuring and Starting a Simple PATHMON Environment (Task 1)	6-1
Restarting the System and Adding Configured TERM Objects (Task 2)	6-5
Communicating With Another PATHMON Environment (Task 3)	6-7
Summary of Example	6-11

7. Overview of PATHCOM

PATHCOM Interface to PATHMON Environments	7-1
PATHMON-Controlled Objects	7-1
Commands and Object States	7-1
Command List	7-2
Command and Object Relationships	7-4
Command Format	7-5
Interactive Mode	7-6
Noninteractive Mode	7-6
PATHMON Configuration File	7-7
Guardian File Names	7-7
File Name Format	7-7
File Name Expansion	7-8

8. Pathway Environment Control Commands

SET PATHWAY Command	8-1
Other Commands	8-4

9. Terminal Control Process (TCP) Commands

ADD TCP Command	9-2
ALTER TCP Command	9-3
CONTROL TCP Command	9-4
DELETE TCP Command	9-9
INFO TCP Command	9-10
PRIMARY TCP Command	9-13
REFRESH-CODE TCP Command	9-14
RESET TCP Command	9-16
SET TCP Command	9-18
SHOW TCP Command	9-32
START TCP Command	9-33
STATS TCP Command	9-35
STATUS TCP Command	9-37
STOP TCP Command	9-42
SWITCH TCP Command	9-44
Wild Card Support for TCP Commands	9-45
Example	9-45

10. TERM Commands

ABORT TERM Command	10-2
ADD TERM Command	10-4
ALTER TERM Command	10-5
CONTROL TERM Commands	10-7
DELETE TERM Command	10-9
INFO TERM Command	10-10
INSPECT TERM Command	10-13
RESET TERM Command	10-15
RESUME TERM Command	10-16
SET TERM Command	10-18
SHOW TERM Command	10-28
START TERM Command	10-30
STATS TERM Command	10-32
STATUS TERM Command	10-35
STOP TERM Command	10-40
SUSPEND TERM Command	10-42
Wild Card Support for TERM Commands	10-43
Example	10-44
Synchronous Upgrade for SCOBOL Applications	10-44

10. TERM Commands (continued)

[Procedure for Synchronous Upgrade](#) 10-45

11. PROGRAM Commands

[ADD PROGRAM Command](#) 11-2

[ALTER PROGRAM Command](#) 11-3

[DELETE PROGRAM Command](#) 11-5

[INFO PROGRAM Command](#) 11-6

[RESET PROGRAM Command](#) 11-8

[RUN PROGRAM Command](#) 11-10

[SET PROGRAM Command](#) 11-15

[SHOW PROGRAM Command](#) 11-24

[Wild Card Support for PROGRAM Commands](#) 11-25

[Example](#) 11-25

12. Tell Message Commands

[DELETE TELL Command](#) 12-2

[INFO TELL Command](#) 12-3

[TELL TERM Command](#) 12-4

13. TCP Messages (Numbers 3000-3999)

[General Information](#) 13-1

[Additional Error Information](#) 13-1

[Operating System Error Numbers](#) 13-2

[SCREEN COBOL Errors](#) 13-2

[TCP Messages](#) 13-2

A. Syntax Summary

B. PATHCOM Reserved Words

[TS/MP Objects](#) B-1

[Pathway/iTS Objects](#) B-3

C. Configuration Limits and Defaults

D. Migration Information

[Interprocess Communication Issues](#) D-1

[Application Conversion Guidelines](#) D-2

E. Setting TMF Parameters

- [SET TERM and SET PROGRAM Commands and TMF](#) E-1
- [Effect of TMF Parameters in PATHCOM on SCREEN COBOL SEND Operations](#) E-1
- [Timeouts on SEND Operations to Servers](#) E-4
- [TCP Checkpointing Strategy](#) E-4
- [Precautions for Using TMF Parameters](#) E-5

F. Setting the DISPLAY-PAGES Parameter

- [Screen Caching](#) F-1
- [Terminal Memory Organization](#) F-1
- [Determining the DISPLAY-PAGES Value](#) F-2
- [Estimating Field-Attribute Entries](#) F-3
- [Assessing Terminal Capacity](#) F-3

G. Source Code for Programs in Section 6

- [Source Code for Screen Program](#) G-1
- [Source Code for Server Program](#) G-5

Index

Examples

- [Example 4-1.](#) [INFO TCP Display](#) 4-4
- [Example 4-2.](#) [INFO TERM Display](#) 4-5
- [Example 4-3.](#) [INFO PROGRAM Display](#) 4-6
- [Example 4-4.](#) [STATUS TCP Display](#) 4-7
- [Example 4-5.](#) [STATUS TCP With STATE Option](#) 4-7
- [Example 4-6.](#) [STATUS TCP With DETAIL Option](#) 4-8
- [Example 4-7.](#) [STATUS TERM Display](#) 4-8
- [Example 4-8.](#) [STATUS TERM With DETAIL Option](#) 4-9
- [Example 4-9.](#) [STATUS TERM With STATE Option](#) 4-9
- [Example 4-10.](#) [STATS TCP Display](#) 4-11
- [Example 4-11.](#) [STATS TCP With DETAIL Option](#) 4-12
- [Example 4-12.](#) [STATS TERM Display](#) 4-13
- [Example 5-1.](#) [Sample TCP Statistics](#) 5-7
- [Example 5-2.](#) [Sample TCP Statistics for POOL INFO](#) 5-8
- [Example 5-3.](#) [Sample TCP Statistics for AREA INFO](#) 5-12
- [Example 5-4.](#) [Sample TCP Statistics for QUEUE INFO](#) 5-15
- [Example 5-5.](#) [Sample Terminal Statistics](#) 5-18
- [Example 5-6.](#) [Sample Terminal Statistics for I/O INFO](#) 5-19

Examples (continued)

Example 5-7.	Sample Terminal Statistics for AREA INFO	5-23
Example 5-8.	Sample Terminal Statistics for RESPONSE TIME INFO	5-24
Example 5-9.	Sample Terminal Statistics With Frequency Distribution Table	5-25
Example 6-1.	TACL Commands in the COLD1 File	6-2
Example 6-2.	PATHCOM Commands in the CONFIG1 File	6-3
Example 6-3.	Sample LOG1 File	6-5
Example 6-4.	TACL Commands in the COOL1 File	6-6
Example 6-5.	TACL Commands in the COLD2 File	6-8
Example 6-6.	PATHCOM Commands in the CONFIG2 File	6-9
Example 6-7.	Logging Output in the LOG2 File	6-10
Example 10-1.	Example STATUS TERM Display With DETAIL During Shutdown	10-39
Example G-1.	Source Code for the Screen Program	G-2
Example G-2.	Source Code for the Server Program	G-5

Figures

Figure 1-1.	Management View of a Pathway Environment	1-7
Figure 1-2.	PATHMON-Controlled Objects Distributed Over Two CPUs	1-9
Figure 1-3.	PATHMON-Controlled Objects Distributed Over Two Nodes	1-10
Figure 1-4.	Transaction Sources	1-14
Figure 1-5.	Communication Between a Local PATHMON Process and an External TCP	1-16
Figure 1-6.	Requester Access to Server Classes	1-19
Figure 1-7.	System Management Tasks	1-20
Figure 2-1.	PATHMON Environment With a Single Application	2-2
Figure 2-2.	PATHMON Environment With Multiple Applications	2-3
Figure 3-1.	Starting a PATHMON-Controlled Object	3-3
Figure 5-1.	TCP Checkpointing	5-6
Figure 6-1.	PATHMON-Controlled Objects Produced by the Example	6-11
Figure 7-1.	PATHCOM Commands and Object States	7-2
Figure E-1.	SEND Operations With TMF	E-3
Figure F-1.	RAM Organization Within a Terminal	F-2

Tables

Table 1-1.	Configuration and Manual Correspondences	1-2
Table 3-1.	Effects of SHUTDOWN2 Options	3-10
Table 3-2.	Effect of STOPMODE and TRANSMODE Registers on Shutdown Operations	3-11
Table 4-1.	Migration Considerations: Pathway/iTS Object Attribute Values	4-26

Tables (continued)

Table 5-1.	TCP Statistics for POOL INFO	5-8
Table 5-2.	TCP Statistics for AREA INFO	5-13
Table 5-3.	TCP Statistics for QUEUE INFO	5-15
Table 5-4.	Terminal Statistics for I/O INFO	5-19
Table 5-5.	Terminal Statistics for AREA INFO	5-23
Table 5-6.	Terminal Statistics for Response Time	5-24
Table 5-7.	Terminal Statistics for Frequency Distribution	5-26
Table 7-1.	PATHCOM Commands	7-3
Table 7-2.	Commands and Objects	7-4
Table C-1.	Global PATHMON Environment Limits	C-1
Table C-2.	Limits and Defaults for Parameters	C-2
Table D-1.	Pathway/iTS Process High-PIN and Low-PIN Support	D-2
Table F-1.	Storage Capacity of 6520 Terminal	F-3
Table F-2.	Storage Capacity of 6526 Terminal	F-3
Table F-3.	Storage Capacity of 6530 Terminal	F-4
Table F-4.	Storage Capacity of TS530 Terminal	F-4

What's New in This Manual

Manual Information

Abstract

This manual describes the management interface to the HP NonStop™ Pathway/iTS product; it is intended for system managers and operators. It provides instructions and guidelines for configuring and controlling the Pathway/iTS objects (those related to terminal control processes (TCPs), terminals, and intelligent devices) and for monitoring the status and performance of those objects. It also provides the syntax and complete descriptions of all PATHCOM commands for Pathway/iTS objects and cause, effect, and recovery information for all TCP error messages.

Product Version

Pathway/iTS 1.1

Supported Release Version Updates (RVUs)

This publication supports J06.03 and all subsequent J-series RVUs and H06.03 and all subsequent H-series RVUs, until otherwise indicated by its replacement publications.

Part Number	Published
426748-005	May 2009

Document History

Part Number	Product Version	Published
426748-001	Pathway/iTS 1.0	October 2000
426748-002	Pathway/iTS 1.0	February 2006
426748-003	Pathway/iTS 1.1	May 2007
426748-004	Pathway/iTS 1.1	May 2008
426748-005	Pathway/iTS 1.1	May 2009

New and Changed Information

Changes in the H06.18/J06.07 manual:

- Added a note for CONTROL TERM commands below:
 - Figure 7-1 PATHCOM Commands and Object States on page [7-2](#)
 - Table 7-1 PATHCOM Commands on page [7-4](#)
- Updated CONTROL command under:
 - Table 7-1 PATHCOM Commands on page [7-3](#)

- Table 7-2 Commands and Objects on page [7-4](#)
- Added [CONTROL TERM Commands](#) on page 10-7.
- Updated considerations of:
 - ABORT TERM command on page [10-3](#)
 - STOP TERM command on page [10-41](#)
- Updated the format of the display returned by the STATUS TERM command on page [10-37](#).
- Updated information about display format of the STATUS TERM command on page [10-38](#).
- Added [Synchronous Upgrade for SCOBOL Applications](#) on page 10-44.

Changes in the H06.14/J06.03 Manual

- Supported release statements have been updated to include J-series RVUs.
- Added the description for:
 - [Wild Card Support for TCP Commands](#)
 - [Wild Card Support for TERM Commands](#)
 - [Wild Card Support for PROGRAM Commands](#)

Changes in the H06.10 Manual

Added error messages, 3246 to 3253 on page [13-56](#).

Product Changes

Pathway/iTS 1.1 provides large message (more than 32000 bytes) communication support between the SCOBOL requesters or converted Java clients and the Pathway servers. The large message communication support is available to the TS/MP Pathsend requesters in the form of Dialog styled communications (SERVERCLASS_DIALOG_* APIs). The TCP component of Pathway/iTS 1.1 is called PATHTCP4, which uses SERVERCLASS_SEND_* APIs for the processing of the SEND verb for SCOBOL requesters and converted Java clients. PATHTCP4 does not communicate to either the owner or external PATHMON process for link management activities. Instead of performing Guardian WRITEREAD on Pathway server processes to communicate with them, PATHTCP4 becomes a Pathsend requester and uses Pathsend APIs to communicate with Pathway servers. Therefore, any existing Pathway/iTS 1.0 application can be used along with Pathway/iTS 1.1 with no or minimal changes to the applications. PATHTCP4 also supports four SCOBOL verbs-DIALOG-ABORT, DIALOG-BEGIN, DIALOG-END, and DIALOG-SEND (for both SCOBOL requesters and converted Java clients) that resemble dialog styled communications in Pathsend requesters. PATHTCP4 uses

SERVERCLASS_DIALOG_* APIs for executing these dialog verbs that enable clients to send more than 32000 bytes message to the same Pathway server process.

The SCUP and Web Client components of Pathway/iTS 1.1 product are changed to support the new functionalities. For compatibility reasons, Pathway/iTS 1.1 package includes the legacy TCP component, that is, PATHTCP3 product.

For further information, see Section 7 of the *Pathway/iTS Web Client Programming Manual*.

Changes in the G06.28 Manual

- Changed the maximum number of concurrently running requester processes from 150 to 800 under [MAXTCPS number](#) on page 8-2.
- Rebranded the terminology in the manual.

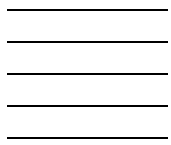
Changes in the G06.27 Manual

The HP NonStop Pathway/iTS product was formerly called Pathway/TS. For the Pathway/iTS 1.0 independent product release, the product was renamed to conform to current Compaq product naming standards and to reflect the new internet (web client) capabilities of the product. After the first reference to the product name in each section of this manual, subsequent references use the shortened form of the name, Pathway/iTS.

Product Changes

- Support for web clients created by converting SCREEN COBOL requesters using the SCREEN COBOL Utility Program (SCUP) CONVERT command. See [Section 1, Introduction to Pathway/iTS System Management](#) and [Browser-Based Web Clients](#) on page 1-14.
- Support for intelligent device support (IDS) requesters that use the sockets protocol. See [Section 1, Introduction to Pathway/iTS System Management](#).
- New gateway program to support web clients.
 - [TCP Objects](#) on page 1-12 provides a general description of the gateway.
 - [Configuring TCPs](#) on page 2-6 describes how to configure TCP objects to use the gateway.
 - [Configuring TERM and PROGRAM Objects](#) on page 2-10 describes how to configure TERM objects to use the gateway.
 - Web gateway-specific information has been added to the descriptions of these TCP configuration attributes: [MAXREPLY bytes](#) on page 9-25, [MAXTERMDATA bytes](#) on page 9-26, [SERVERPOOL bytes](#) on page 9-28, [TERMBUF bytes](#) on page 9-29, and [TERMPPOOL bytes](#) on page 9-30.
- New router process to support web clients and sockets IDS requesters

- [Router Processes](#) on page 1-13 provides a general description of the router process.
- [Starting Router Processes](#) on page 3-1 describes how to start router processes.
- [Stopping Router Processes](#) on page 3-7 describes how to stop router processes.
- New TCP configuration attribute, SENDMSGTIMEOUT, to support filtering of 3161 timeout error messages when the ON ERROR clause is used with a SEND MESSAGE statement. [SET TCP Command](#) on page 9-18 describes this new attribute, and the syntax descriptions of the other TCP configuration commands include the attribute.
- Eleven new TCP error messages, numbered [3140](#) through [3150](#), have been added.



About This Manual

This manual is both a task-oriented manual and a reference manual. It provides instructions and guidelines for configuring and controlling the HP NonStop Pathway/iTS objects in a PATHMON environment and for monitoring the status and performance of those objects. The Pathway/iTS objects are those that operate under the run-time portions of Pathway/iTS (the terminal control process (TCP) and SCREEN COBOL run-time environment). It also provides syntax and complete descriptions of all PATHCOM commands for Pathway/iTS objects, in addition to cause, effect, and recovery information for TCP error messages.

The *TS/MP System Management Manual* is a prerequisite to this manual. For information about management of the PATHMON environment as a whole and management of TS/MP objects, see *TS/MP System Management Manual*.

Who Should Read This Manual

This manual is intended for those individuals responsible for starting, configuring, and managing the Pathway/iTS objects in a PATHMON environment, using the PATHCOM interactive interface. It is assumed that readers have a general knowledge of HP NonStop programming concepts.

The task-oriented parts of this manual are also intended for those individuals writing programs to manage the Pathway/iTS objects in a PATHMON environment programmatically. Such programmers also need the reference information in the *Pathway/iTS Management Programming Manual* and the *TS/MP Management Programming Manual*.

Related Documentation

In addition to this manual, information about Pathway/iTS appears in these publications:

<i>Pathway/iTS SCREEN COBOL Reference Manual</i>	Describes the SCREEN COBOL programming language which is used for writing programs that define and control terminal displays or intelligent devices for online transaction processing applications running in a PATHMON environment.
<i>Pathway/iTS SCUP Reference Manual</i>	Describes managing a SCREEN COBOL library with the SCREEN COBOL Utility Program (SCUP).
<i>Pathway/iTS Web Client Programming Manual</i>	Describes how to convert SCREEN COBOL requesters to web clients, explains how to build and deploy those clients, and also provides the information Java developers and web designers need to to modify and enhance the Java and HTML portions of the converted clients.
<i>Pathway/iTS TCP and Terminal Programming Guide</i>	A guide for programmers who are writing SCREEN COBOL requesters to be used in Pathway applications.

<i>Pathway/iTS Management Programming Manual</i>	Describes the management programming interface for Pathway/iTS objects in the PATHMON environment.
<i>Pathway Products Glossary</i>	Defines technical terms used in this manual and in other manuals for the Pathway products: Pathway/iTS, TS/MP, and Pathway/XM.
<i>Pathway/XM System Management Manual</i>	Describes the higher-level management interface provided by the Pathway/XM product, which you might want to use to configure and manage your Pathway environment rather than use the management interfaces provided by Pathway/iTS and NonStop TS/MP.
<i>Operator Messages Manual</i>	Describes all messages that are distributed by the Event Management Service (EMS), including those generated by NonStop TS/MP and Pathway/iTS processes.
<i>TS/MP Pathsend and Server Programming Manual</i>	Describes the Pathsend procedure calls, included as part of the TS/MP product, and how to use those calls to write requester programs. It also describes how to design and code Pathway servers for use with all types of requesters and clients.

Notation Conventions

General Syntax Notation

The following list summarizes the notation conventions for syntax presentation in this manual.

UPPERCASE LETTERS. Uppercase letters indicate keywords and reserved words; enter these items exactly as shown. Items not enclosed in brackets are required. For example:

MAXATTACH

lowercase italic letters. Lowercase italic letters indicate variable items that you supply. Items not enclosed in brackets are required. For example:

file-name

[] Brackets. Brackets enclose optional syntax items. For example:

TERM [*\system-name.*] *\$terminal-name*

INT [ERRUPTS]

A group of items enclosed in brackets is a list from which you can choose one item or none. The items in the list may be arranged either vertically, with aligned brackets on

each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
LIGHTS [ ON
        [ OFF
        [ SMOOTH [ num ] ]
```

```
K [ X | D ] address-1
```

{ } Braces. A group of items enclosed in braces is a list from which you are required to choose one item. The items in the list may be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LISTOPENS PROCESS { $appl-mgr-name
                   { $process-name }
```

```
ALLOWSU { ON | OFF }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
INSPECT { OFF | ON | SAVEABEND }
```

... Ellipsis. An ellipsis immediately following a pair of brackets or braces indicates that you can repeat the enclosed sequence of syntax items any number of times. For example:

```
M address-1 [ , new-value ]...
```

```
[ - ] { 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 }...
```

An ellipsis immediately following a single syntax item indicates that you can repeat that syntax item any number of times. For example:

```
"s-char..."
```

Punctuation. Parentheses, commas, semicolons, and other symbols not previously described must be entered as shown. For example:

```
error := NEXTFILENAME ( file-name ) ;
```

```
LISTOPENS SU $process-name.#su-name
```

Quotation marks around a symbol such as a bracket or brace indicate the symbol is a required character that you must enter as shown. For example:

```
"[ repetition-constant-list ]"
```

Item Spacing. Spaces shown between items are required unless one of the items is a punctuation symbol such as a parenthesis or a comma. For example:

```
CALL STEPMOM ( process-id ) ;
```

If there is no space between two items, spaces are not permitted. In this example, there are no spaces permitted between the period and any other items:

```
$process-name.#su-name
```

Line Spacing. If the syntax of a command is too long to fit on a single line, each continuation line is indented three spaces and is separated from the preceding line by a blank line. This spacing distinguishes items in a continuation line from items in a vertical list of selections. For example:

```
ALTER [ / OUT file-spec / ] CONTROLLER
      [ , attribute-spec ]...
```

Notation for Messages

The following list summarizes the notation conventions for the presentation of displayed messages in this manual.

Nonitalic text. Nonitalic letters, numbers, and punctuation indicate text that is displayed or returned exactly as shown. For example:

Backup Up.

lowercase italic letters. Lowercase italic letters indicate variable items whose values are displayed or returned. For example:

p-register
process-name

[] Brackets. Brackets enclose items that are sometimes, but not always, displayed. For example:

```
Event number = number [ Subject = first-subject-value ]
```

A group of items enclosed in brackets is a list of all possible items that can be displayed, of which one or none might actually be displayed. The items in the list might be arranged either vertically, with aligned brackets on each side of the list, or horizontally, enclosed in a pair of brackets and separated by vertical lines. For example:

```
LDEV ldev [ CU %ccu | CU %... ] UP [ (cpu,chan,%ctlr,%unit) ]
```

{ } Braces. A group of items enclosed in braces is a list of all possible items that can be displayed, of which one is actually displayed. The items in the list might be arranged either vertically, with aligned braces on each side of the list, or horizontally, enclosed in a pair of braces and separated by vertical lines. For example:

```
LBU { X | Y } POWER FAIL
process-name State changed from old-objstate to objstate
{ Operator Request. }
{ Unknown. }
```

| Vertical Line. A vertical line separates alternatives in a horizontal list that is enclosed in brackets or braces. For example:

```
Transfer status: { OK | Failed }
```

% Percent Sign. A percent sign precedes a number that is not in decimal notation. The %pnotation precedes an octal number. The %Bnotation precedes a binary number. The %Hnotation precedes a hexadecimal number. For example:

%005400

P=%p-register E=%e-register

Change Bar Notation

Change bars are used to indicate substantive differences between this manual and its preceding version. Change bars are vertical rules placed in the right margin of changed portions of text, figures, tables, examples, and so on. Change bars highlight new or revised information. For example:

The message types specified in the REPORT clause are different in the COBOL85 environment and the Common Run-Time Environment (CRE).

The CRE has many new message types and some new message type codes for old message types. In the CRE, the message type SYSTEM includes all messages except LOGICAL-CLOSE and LOGICAL-OPEN.

Introduction to Pathway/iTS System Management

This manual provides the information you need to define and control objects under the HP NonStop Pathway/iTS product to support SCREEN COBOL requesters and web clients in a Pathway environment by using the PATHCOM interface. It also provides syntax and complete descriptions of all PATHCOM commands for Pathway/iTS objects, in addition to cause, effect, and recovery information for error messages reported by Pathway/iTS terminal control processes (TCPs).

Objects defined and controlled under TS/MP, which support server processes and the Pathway environment as a whole, are described in a separate manual. For more information about creating and managing server objects and TS/MP, see the *TS/MP System Management Manual*.

As an alternative to using PATHCOM, the Pathway/XM product is available. Pathway/XM simplifies the management of Pathway applications by means of several features. It provides a centralized configuration and management interface with automatic load balancing, online reconfiguration and replacement of servers, simplified object naming, and an increased overall system capacity. This manual does not cover how to manage a Pathway environment through Pathway/XM. For more information, see the *Pathway/XM System Management Manual*.

You can also manage the Pathway environment programmatically by using the Pathway/iTS and TS/MP management programming interface. This interface lets you write management applications that automate configuration and control tasks. The interface is a token-oriented interface based on the Subsystem Programmatic Interface (SPI) message format, procedures, and definition files. For more information about the Pathway management programming interface, see the *Pathway/iTS Management Programming Manual* and *TS/MP Management Programming Manual*.

Supported Release Version Updates (RVUs)

Pathway/iTS 1.0 is a release-independent product that can be used with these RVUs of the HP NonStop operating system:

- D42.00 and later D-series RVUs
- G02.00 and later G-series RVUs
- H06.03 and later H-series RVUs

TS/MP is required by Pathway/iTS. HP NonStop Transaction Management Facility is also required if you are using transaction statements in your requester programs.

Which Sections Do You Need?

This manual is organized into logical groups of information for easy reference. [Table 1-1](#) is a descriptive map showing which sections are relevant to particular operating environments and tasks.

Table 1-1. Configuration and Manual Correspondences (page 1 of 3)

If Your Configuration Includes...	You Need...	To Perform the Following...
Any Pathway/iTS product components	Section 1	Understand the architecture of the Pathway environment, including TS/MP, the Pathway/iTS product, and related products
	Section 2	Configure limits for Pathway/iTS objects
	Section 2	Specify node independence for Pathway/iTS objects
	Section 2	Configure terminal control process (TCP) objects
	Section 3	Start and stop Pathway/iTS objects
	Section 4	Display information about the configuration and status of Pathway/iTS objects
	Section 4	Reconfigure Pathway/iTS objects
	Section 4	Monitor exception conditions for Pathway/iTS objects
	Section 4	Improve the performance of Pathway/iTS objects
	Section 4	Report problems to your service provider
	Section 4	Send messages to terminal users
	Section 4	Migrate Pathway/iTS objects to another system
	Section 5	Collect and display statistical information about Pathway/iTS objects
	Section 5	Tune your system using statistical information
	Section 6; Appendix G	See an example of configuring and starting a simple PATHMON environment
	Section 7	Understand how to use the PATHCOM interface to manage a PATHMON environment with Pathway/iTS objects
	Section 8	Look up detailed syntax of PATHCOM SET PATHWAY command parameters specific to Pathway/iTS objects

Table 1-1. Configuration and Manual Correspondences (page 2 of 3)

If Your Configuration Includes...	You Need...	To Perform the Following...
Any Pathway/iTS product components	Section 8	Look up how specific TS/MP commands affect Pathway/iTS objects
	Section 9	Look up detailed syntax of PATHCOM commands for TCP objects
	Section 12	Look up detailed syntax of PATHCOM commands for messages sent to terminal users (TELL objects)
	Section 13	Handle error and informative messages from TCPs
	Appendix A	Look up abbreviated syntax of any PATHCOM command for <i>TS/MP</i> or Pathway/iTS objects
	Appendix B	See a list of all PATHCOM reserved words for TS/MP and Pathway/iTS
	Appendix C	Look up limits and default values for Pathway/iTS configuration parameters
Input-output devices and processes such as terminals, intelligent devices, and browser-based web clients	Appendix D	Get information about use of high-PIN processes with the Pathway/iTS product
	Section 2	Configure TERM objects
	Section 10	Look up detailed syntax of PATHCOM commands for TERM objects
Templates for SCREEN COBOL programs that run on multiple input-output devices and processes	Appendix F	Calculate the value of the DISPLAY-PAGES parameter for TERM objects
	Section 2	Configure PROGRAM objects
	Section 11	Look up detailed syntax of PATHCOM commands for PROGRAM objects

Table 1-1. Configuration and Manual Correspondences (page 3 of 3)

If Your Configuration Includes...	You Need...	To Perform the Following...
	Appendix F	Calculate the value of the DISPLAY-PAGES parameter for PROGRAM objects
Multiple PATHMON processes	Section 2	Manage communication between PATHMON environments
The HP NonStop Transaction Management Facility (TMF)	Appendix E	Determine the effect on Pathway/iTS objects of the TMF parameters you configure in your PATHMON environment

Pathway Environment Overview

The Pathway environment includes a group of related software tools that enable your organization to develop, install, and manage online transaction processing applications.

Transaction processing applications consist of two types of programs: requester programs (called clients in other environments) and server programs. Among other benefits, this design allows application logic to be distributed near the resources it manages. For example, presentation services are located in requester programs near the I/O devices; database logic resides in server programs near the database. Requesters and servers communicate using the message system provided by the NonStop operating system.

Depending on your hardware and software configuration, a Pathway environment includes one of these:

- TS/MP (for managing server-based objects and the overall Pathway environment), other software (for managing requester-based objects), and additional optional software.
- Both TS/MP (for managing server-based objects and the overall Pathway environment), the Pathway/iTS product (for managing requester-based objects), and additional optional software.

Your Pathway environment also includes processes and objects you create. The key elements of a Pathway environment are listed in these paragraphs. Each element is described in detail later in this section.

Objects and Processes Provided by TS/MP

The following elements of a Pathway environment are provided by TS/MP. For information about these items, see the *TS/MP System Management Manual*.

- **PATHMON**—the process that manages TCP, TERM, PROGRAM, and SERVER objects. The PATHMON process and the objects controlled by it make up the PATHMON environment. Each PATHMON environment includes only one PATHMON process.
- **PATHCOM**—the interactive interface to the PATHMON process. As an alternative to using PATHCOM, you can write a management application program, using the Subsystem Programmatic Interface (SPI).
- **SERVER** objects—definitions of server classes (multiple copies of server programs). An executing server program is called a server process. You use the PATHMON process to configure and manage Guardian server programs, which reside in the Guardian operating environment, and HP NonStop Open System Services (OSS) server programs, which reside in the OSS operating environment.
- **LINKMON**—the process that manages communications between Pathsend processes and server classes. (Pathsend processes are user applications that use Pathsend procedure calls, rather than the SCREEN COBOL language, to send requests to server classes.)

Objects and Processes Provided by Pathway/iTS

The following elements of a Pathway environment are provided by the Pathway/iTS product.

- **TCP** objects—terminal control processes that perform these functions:
 - execute screen (SCREEN COBOL) programs in your application
 - coordinate communication between screen programs, input-output devices or processes, server processes, and the PATHMON process
 - coordinate communication between router processes and web clients
 - coordinate communication between router processes and intelligent devices that use the sockets protocol
- **TERM** objects—definitions of tasks that use screen programs to control input-output devices such as terminals and workstations, or input-output processes that enable users to communicate with a Pathway application.
- **PROGRAM** objects—templates for creating and starting temporary TERM objects.
- **Router** processes—processes that provide link management and load balancing for web clients and for intelligent devices that use the sockets protocol.

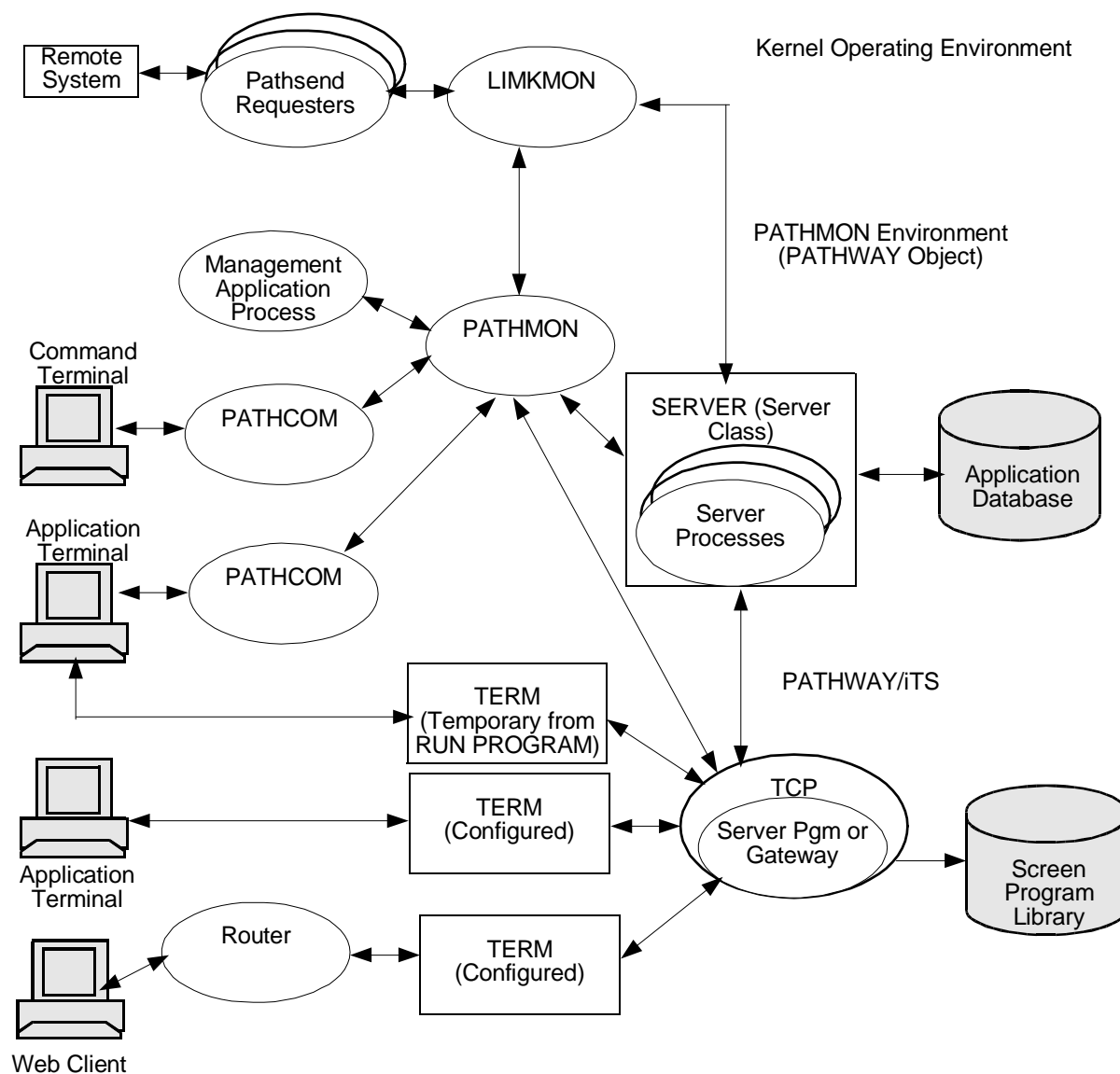
Information about defining and managing these items is given in this manual. However, the router processes are not managed objects within the Pathway environment. If you are using PATHCOM or the management programming interface to configure and manage your Pathway environment, you must start the router processes in the Guardian environment (for example, through the HP Tandem Advanced Command Language (TACL)) and only limited management capabilities are

available. The Pathway/XM product allows you to configure router processes as part of a Pathway/XM environment.

Pathway Environment Configurations

Depending on your configuration, a Pathway environment might also include other software, such as TMF and the run-time portion of the HP NonStop Remote Server Call/MP (RSC/MP) product.

[Figure 1-1](#) on page 1-7 shows a system manager's view of a simple Pathway environment, including the PATHMON process and its TS/MP and Pathway/iTS objects; a command terminal running PATHCOM; a management application process; and a set of application terminals.

Figure 1-1. Management View of a Pathway Environment

For information about configuring Pathway applications in more complex environments—for example, including Open System Services (OSS) servers—see the *TS/MP System Management Manual*.

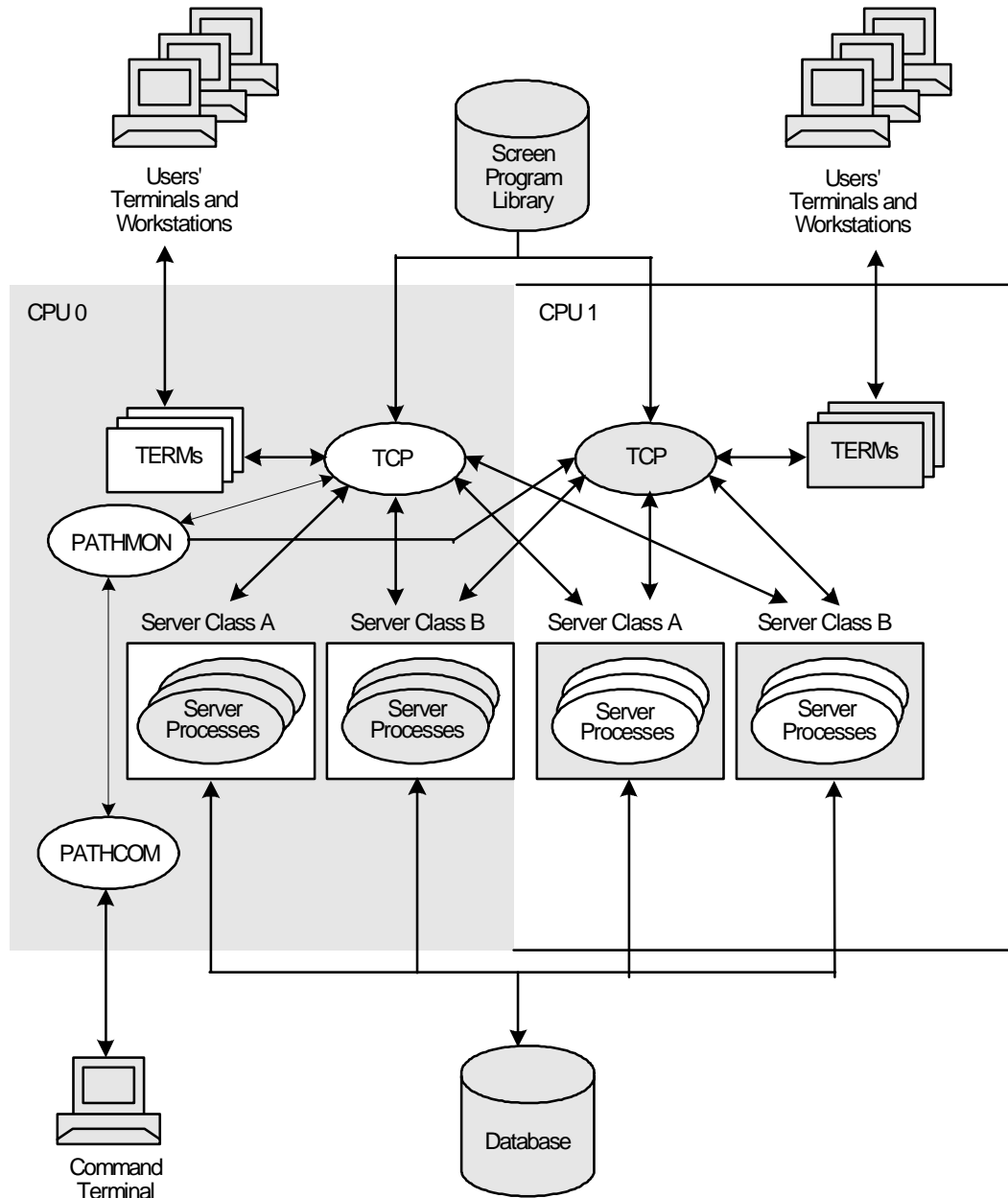
Distributing a Pathway Environment

Depending on the requirements of your application, you might distribute a Pathway environment over several CPUs in a single HP NonStop system or among several NonStop systems.

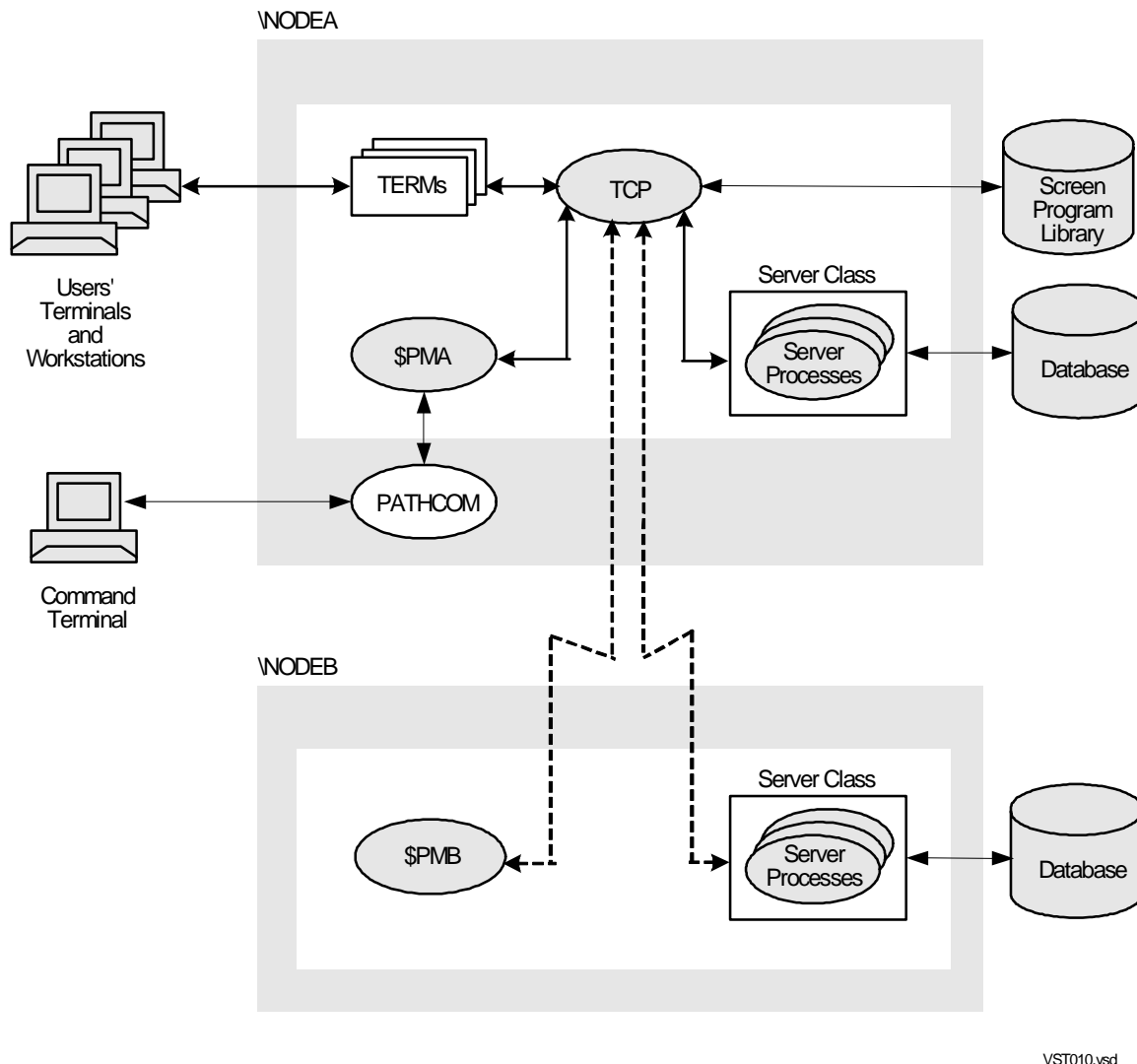
A PATHMON environment is a set of Pathway environment objects controlled through a single PATHMON process. You arrange PATHMON-controlled objects in different processors of your NonStop system to support the workload required by your application and to achieve optimum response times for your business transactions. [Figure 1-2](#) on page 1-9 shows a single PATHMON environment distributed over two CPUs in a system.

You can distribute functions over different physical or geographical locations by distributing a single PATHMON environment across a network of nodes connected by communications lines, or by configuring two or more PATHMON environments to communicate across such a network. In such distributed environments, you can enable TCPs on one node to access a database maintained by server processes on another node. This configuration allows multiple applications on different nodes to share data.

[Figure 1-3](#) on page 1-10 shows two PATHMON environments distributed over two nodes. A TCP in node \NODEA forms a link across a communications line to a server class controlled by the PATHMON process on \NODEB. The TCP is an external TCP with respect to the objects controlled by \$PMB.

Figure 1-2. PATHMON-Controlled Objects Distributed Over Two CPUs

VST009.vsd

Figure 1-3. PATHMON-Controlled Objects Distributed Over Two Nodes

VST010.vsd

TS/MP Objects and Processes

The following elements of a Pathway environment are provided by TS/MP. For detailed information about these items, see the *TS/MP System Management Manual*.

PATHMON Object

The PATHMON object represents the PATHMON process, the main control process in a PATHMON environment. Each PATHMON environment has only one PATHMON process.

The PATHMON process receives information about and communicates with all of the objects under its control. The PATHMON process creates and maintains a file in which it stores the configuration information for each of its objects.

The PATHMON process enforces the limits you set for the environment and monitors the operation of the objects under its control by:

- Keeping a record of the object definitions in the PATHMON configuration file
- Starting TCPs, TERM objects, PROGRAM objects, and SERVER processes
- Facilitating access between TCPs or LINKMON processes and server processes to support requester/server communication
- Reporting status information and system errors about PATHMON-controlled objects
- Shutting down all or part of a PATHMON environment by stopping individual objects
- Automatically restarting failed objects

PATHCOM Processes

You can use PATHCOM at a command terminal to communicate interactively with the PATHMON process. Other users use PATHCOM at application terminals to start and run Pathway applications. Multiple PATHCOM processes can communicate with a single PATHMON process concurrently, each supporting a particular user.

SERVER Objects

The SERVER object represents a server class, which is a group of server processes that perform a specific type of application work (for example, adding customer names and addresses, calculating invoice totals, or checking inventory). The PATHMON process controls two types of server classes: Guardian server classes residing in the Guardian environment and OSS server classes residing in the OSS environment.

The PWY2TUX translation server for the HP NonStop TUXEDO system is a special-purpose OSS server that enables SCREEN COBOL requesters to access NonStop TUXEDO application services. For more information about this translation server, see the *NonStop TUXEDO System Pathway Translation Servers Manual*.

LINKMON Processes

A LINKMON process is a multitasking process that handles the interface to Pathway servers from requesters other than SCREEN COBOL requesters. Such requesters include Pathsend requesters; workstation clients that use RSC/MP; and NonStop TUXEDO clients.

Pathsend requesters (also called Pathsend processes) are requesters written in languages other than SCREEN COBOL that use the Pathsend procedure calls to access Pathway server classes. RSC/MP workstation clients communicate with Pathway servers indirectly by means of Pathsend requesters created by RSC/MP on behalf of the clients.

Pathway/iTS Objects and Processes

The following elements of a Pathway environment are provided by the Pathway/iTS product. Information about defining and managing these items is given in this manual.

TCP Objects

A terminal control process (TCP) interprets and executes screen program instructions for each input-output (I/O) device or process the TCP is configured to handle. The TCP coordinates communication between screen programs and their I/O devices or processes and, with the help of the PATHMON process, establishes links between the screen programs and server processes. The TCP also performs these tasks:

- Verifies and stores terminal context data and terminal control logic
- Gathers statistics about TERM objects, the server processes, and itself
- Reports operating errors and status information to the PATHMON process
- Provides integrity across failure situations

Because it is a multithreaded process, a TCP can concurrently manage many user terminals and execute many screen programs. To accomplish these tasks, the TCP runs its TERM or PROGRAM objects as threads, interleaving the concurrent processing of requests originating from many I/O devices or processes. This multithreading allows the TCP to handle complex groups of operations for many users at the same time.

The TCP manages the operation of multiple I/O devices by maintaining separate screen program code and data areas for each device under its control. In addition to managing operation of terminals, the TCP supports card readers, local printers, and bar-code readers when they are attached to terminals.

To support browser-based web clients, Pathway/iTS also provides a prewritten screen program called the gateway. When you configure a TCP to interpret and execute the gateway program, the gateway threads work with the router process to manage links for web clients. Each TERM object under this TCP is associated with a socket, and there is one client connection per TERM object. Each gateway requester thread is associated with one router process, whose connections it handles. The TERM objects under this TCP are all associated with a particular router process and application.

TERM and PROGRAM Objects

TERM objects represent tasks controlling the input-output devices and processes that allow users to interact directly or programmatically with a Pathway application. Each task runs as a thread in the TCP; the TCP can handle many such threads concurrently. There are two types of these objects: configured TERM objects and temporary TERM objects.

Configured TERM objects are those you define and add to the PATHMON configuration file. Temporary TERM objects are created for you (and automatically

deleted) by the PATHMON process in response to a RUN PROGRAM command, using a template you configure as a PROGRAM object. (A PROGRAM object is simply a mechanism for creating temporary TERM objects as they are needed.)

Router Processes

A router process listens and distributes connection requests from web clients and from intelligent devices that use the raw sockets protocol. A router process works together with a TCP that is interpreting and executing either the Pathway/iTS gateway process (for web clients) or a user-written IDS requester (for sockets intelligent devices). Each router process is configured with a TCP/IP port used to associate an application with connection requests.

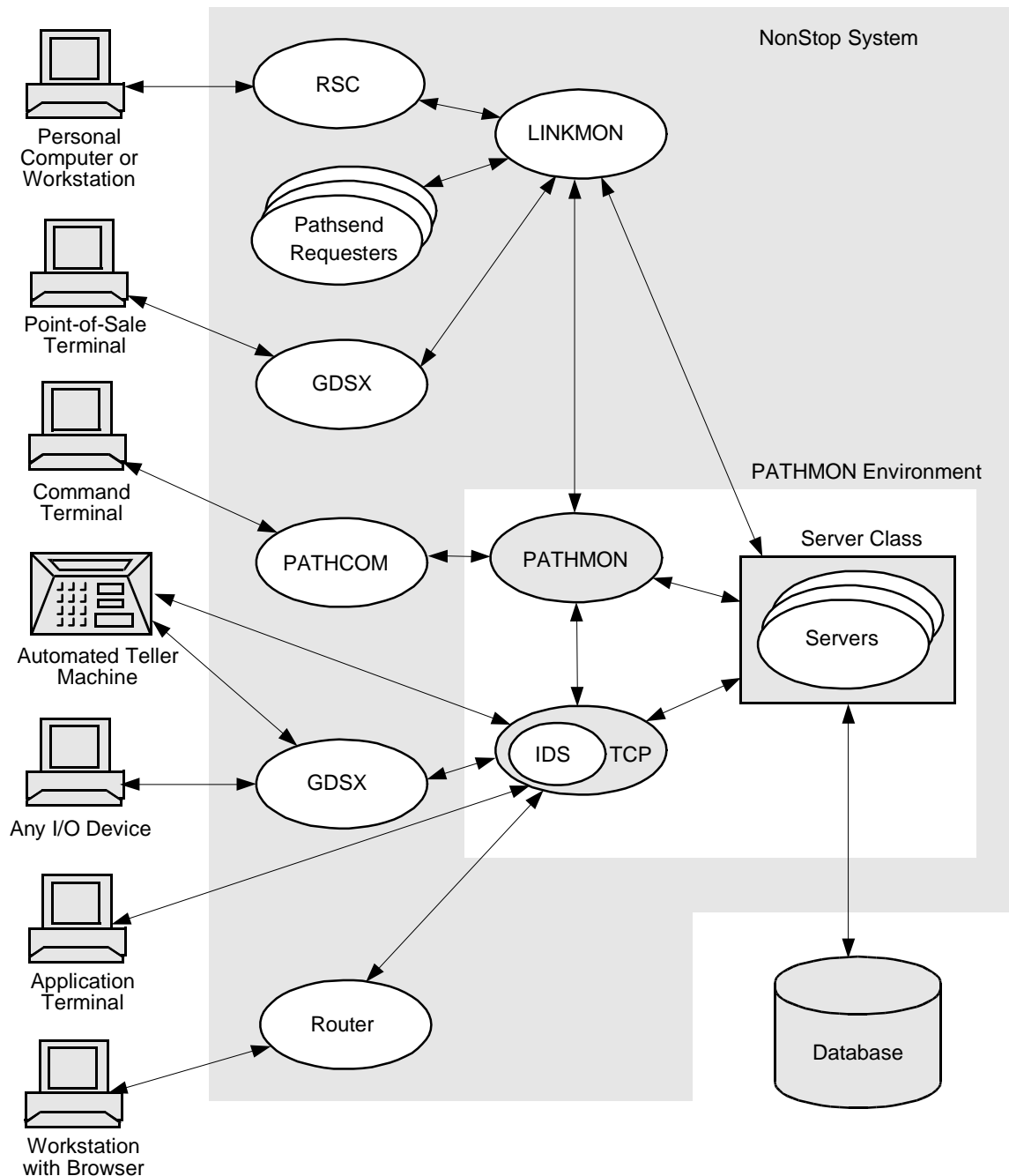
The router processes distribute connection requests from web clients or intelligent devices to started TERM objects within the TCPs in such a way that the connection load is optimally balanced among all the TCPs in the Pathway environment. When all the TERM objects in all the configured TCPs are in use, the router queues new connection requests until a TERM object is available. The router also queues the TERM objects that are ready for a connection until a connection request arrives.

Transaction Sources

A variety of devices and processes may interact with the PATHMON process and its objects to supply required resources and services or to make requests. These include PATHCOM or a management application process, in addition to terminals controlled by TCP objects. In addition, these devices and processes might include:

- Browser-based web clients
- Personal computers and workstations (using the RSC/MP product)
- External TCP objects, that is, terminals running outside of your PATHMON environment
- Intelligent devices such as an automated teller machine (using intelligent device support (IDS))
- SNA devices (using the SNAX High-Level Support (SNAX/HLS) product)
- Unsupported or special-function I/O devices (using the Extended General Device Support (GDSX) product)
- Pathsend processes (using Pathsend procedure calls to communicate with the LINKMON process)

[Figure 1-4](#) on page 1-14 shows examples of devices that interact with PATHMON-controlled objects, either directly or through the LINKMON process. Note that the LINKMON, PATHMON, and PATHCOM processes are provided by TS/MP. Server processes are defined and managed under TS/MP. For information on these processes, see the *TS/MP System Management Manual*.

Figure 1-4. Transaction Sources

VST007.vsd

Browser-Based Web Clients

Pathway/iTS provides the capability to convert SCREEN COBOL programs to web clients consisting of Java code and HTML pages, and to deploy those clients in a Pathway environment. The Pathway/iTS router process provides link management and load balancing for these web clients. For information about creating and deploying these clients, see the *Pathway/iTS Web Client Programming Manual*.

Personal Computers and Workstations

Personal computers (PCs) and workstations can access a PATHMON environment using the RSC/MP product.

RSC/MP enables client-server computing by supporting a variety of hardware and software configurations and communications protocols for personal computers (PCs) and workstations. In addition to delegating some processing normally performed by the NonStop system to the PC, RSC/MP allows you to take advantage of the graphical user interfaces (GUIs)—pull-down menus, icons, dialog boxes, and online help—that are available on the PC.

For more information about the RSC/MP product, see the *HP NonStop Remote Server Call (RSC/MP) Installation and Configuration Guide*.

External TCPs

An external TCP is a TCP running outside of your PATHMON environment. The external TCP is controlled by another PATHMON process, but it can communicate with processes within your PATHMON environment. You determine how many (if any) external TCPs can communicate with your PATHMON environment when you configure it.

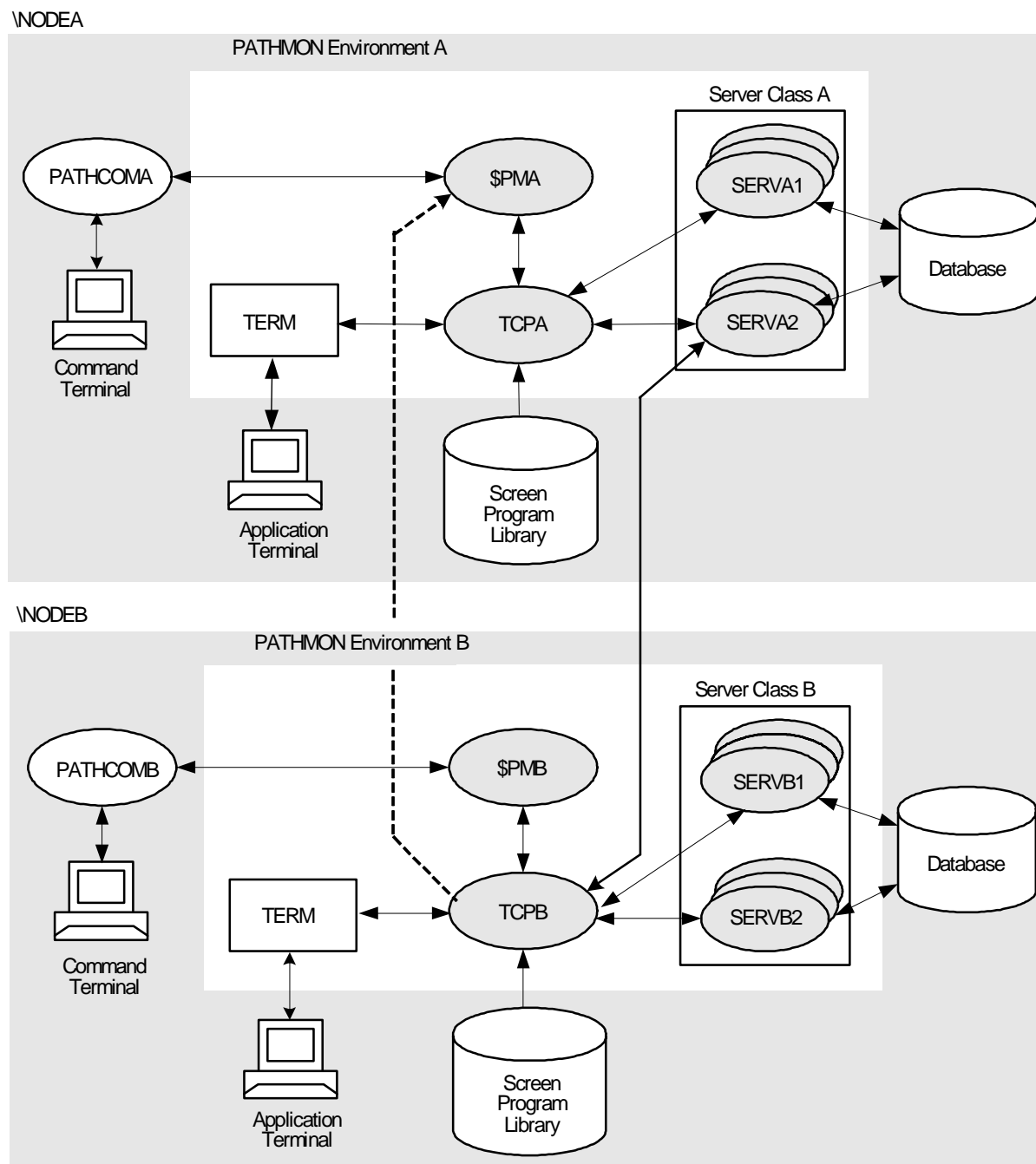
As an example, a screen program run by an external TCP might need to communicate with a server class in your PATHMON environment. The external TCP communicates with the PATHMON process in your environment to request a link to a server class in your environment. This situation arises when:

- PATHMON environments are configured along application lines. For example, suppose that Application A is running under PATHMON process A, and Application B is running under PATHMON process B, and the screen program in Application A needs to use the server class in Application B.
- PATHMON environments are configured along network node boundaries. For example, suppose that a PATHMON environment on the node named \SANFRAN supports an order-processing application for the San Francisco area; a PATHMON environment on the node named \WHOUSE supports an inventory application for a central warehouse; the order-processing application needs information about inventory items.

[Figure 1-5](#) on page 1-16 illustrates a PATHMON environment (running under PATHMON process \$PMA) communicating with another PATHMON environment (running under PATHMON process \$PMB). \$PMA accepts requests from the TCP on PATHMON environment B (TCPB) and grants TCPB links to the server class that \$PMA controls. Then, TCPB opens a server process in \$PMA's system and sends transaction requests to that server process. Conversely, TCPA can request server-class links from \$PMB. PATHMON environment A views TCPB as an external TCP.

The term “external TCP” is relative to your point of view: a TCP that runs in another PATHMON environment is external to your environment, but it is local to that other environment. In the other environment, that TCP is exactly like any other local TCP.

Figure 1-5. Communication Between a Local PATHMON Process and an External TCP



VST005.vsd

Intelligent Devices

Intelligent devices—such as automated teller machines (ATMs), workstations, bar-code readers, and machines running the Unix operating system—can access Pathway server classes through SCREEN COBOL requester programs and the TCP.

This access route is possible when you use the intelligent device support (IDS) facility, which is part of the TCP. By delegating some processing to the intelligent device, IDS makes better use of the device's processing ability and reduces the workload of the NonStop system. For Unix machines and other intelligent devices that use the sockets protocol, the Pathway/iTS router process provides link-management and load-balancing functions.

SNA Devices

Requests from SNA devices can be handled by SNAX High-Level Support (SNAX/HLS), which allows Pathway application programs to communicate with SNA devices and host software. Using SNAX/HLS, Pathway applications can:

- Access IBM host software products (such as CICS and IMS) for distributed transaction processing
- Communicate with intelligent SNA controllers such as the IBM 3600 or 4700 Financial Subsystems, or the IBM 3650 Retail Subsystem

The application interface to SNAX/HLS requires little detailed knowledge of SNAX or SNA. For more information about the SNAX/HLS product, see the *SNAX/HLS Configuration and Control Manual*.

Unsupported or Special-Function I/O Devices

Requests from unsupported or special-function I/O devices can be handled by the Extended General Device Support (GDSX) product. GDSX is designed to help you develop a front-end process that translates requests into a format supported by the TCP or the LINKMON process.

Because GDSX is a complex product, you should not use it to implement an application if the PATHMON process and the TCP can handle the application requirements. However, you might want to use a GDSX process as a front end to a Pathway application if:

- The specified data communications protocols are not supported by the TCP or the LINKMON process.
- Performance is critical. Interpreted SCREEN COBOL might not be efficient enough when used to program a data communications protocol.
- Simplicity of programming is critical. Handling a data communications protocol can be awkward with SCREEN COBOL, which has a limited set of data types and verbs.

- A file or any other device has to be accessed before sending data to the SCREEN COBOL program, for example, when data requires encryption or decryption.

Using the GDSX product, you can implement message switching, develop and modify data communications protocols, and perform data-stream conversions, all of which facilitate access to Pathway server classes. For more information about the GDSX product, see the *Extended General Device Support (GDSX) Manual*.

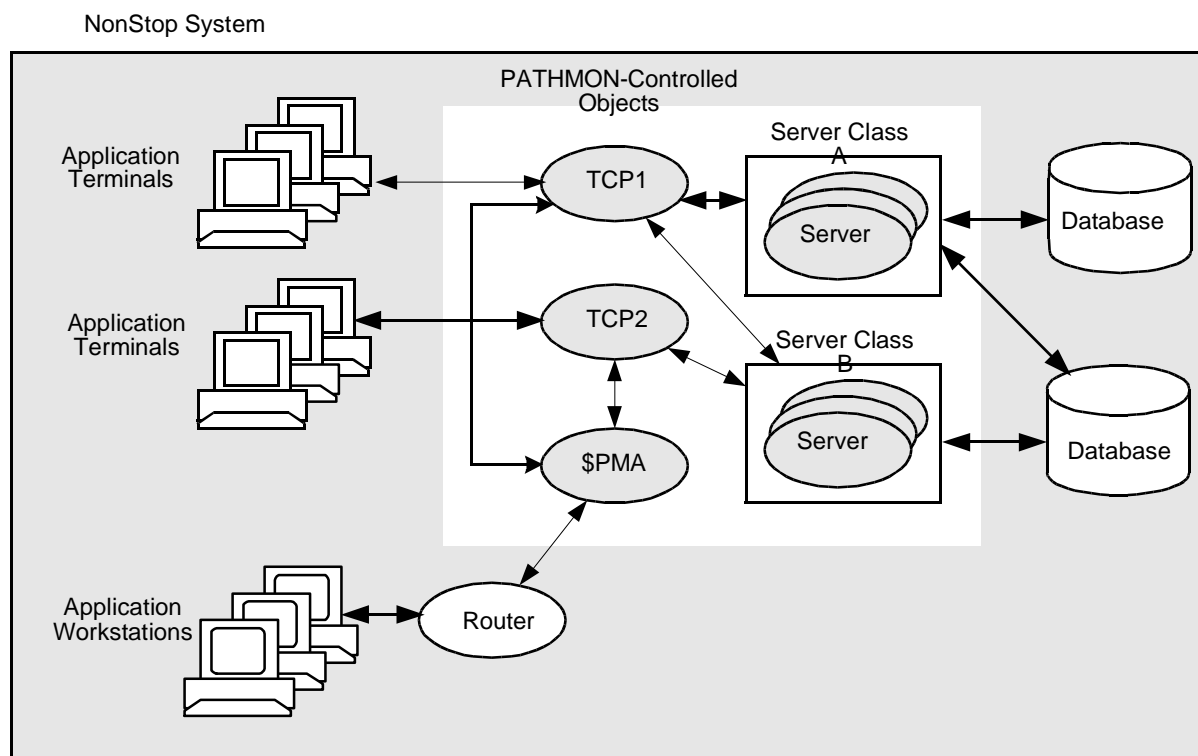
The LINKMON Process

A LINKMON process is a multitasking process that handles the interface to Pathway servers from requesters other than SCREEN COBOL requesters. Such requesters include Pathsend requesters; workstation clients that use RSC/MP; and NonStop TUXEDO clients.

For information about managing LINKMON and Pathsend processes, see the *TS/MP System Management Manual*. For information about how to write Pathsend programs, see the *TS/MP Pathsend and Server Programming Manual*.

Requester-Server Communication

[Figure 1-6](#) on page 1-19 shows a PATHMON environment that includes SCREEN COBOL requesters and web clients. The TCPs, TCP1 and TCP2, interpret multiple SCREEN COBOL requester threads, one per application terminal, and provide link-management functions. TCP2 also executes multiple web gateway threads, one per workstation running a browser and web client, and coordinates with the link-management functions of the router process.

Figure 1-6. Requester Access to Server Classes

VST006.vsd







System Management Tasks

The specific tasks to set up and manage PATHMON-controlled objects in a PATHMON environment depend on your applications, but generally include the tasks shown in [Figure 1-7](#) on page 1-20. You can perform the required tasks by using PATHCOM or by writing a management application, as described further in the *TS/MP System Management Manual*.

Note. [Figure 1-7](#) on page 1-20 describes only PATHMON-controlled objects. If you are using web clients converted from SCREEN COBOL programs or IDS requesters that use the sockets protocol, you should start the router processes before performing these steps and stop the router processes after you shut down the Pathway environment. You should also start and stop any other required processes for other products, such as RSC/MP, as described in the manuals for those products.

Note. Most of the tasks listed in [Figure 1-7](#) on page 1-20 are performed through TS/MP. For complete information on starting a PATHMON environment and managing PATHMON-controlled objects other than TCP, TERM, and PROGRAM objects, see the *TS/MP System Management Manual*.

Figure 1-7. System Management Tasks

Start Pathway Environment		Start PATHMON and PATHCOM Processes
Configure and Start PATHMON Environment		Configure PATHMON Process Specify Global Limits Start PATHMON Environment
Configure PATHMON-Controlled Objects		Configure and Add TCPs Configure and Add Terminals Configure and Add Programs Configure and Add Server Classes
Start PATHMON-Controlled Objects		Start TCPs, Terminals, and Server Classes Run Process Threads for Programs
Maintain PATHMON Environment		Monitor Status and Performance Monitor Exception Conditions Send Messages to Terminal Users Reconfigure When Necessary
Stop PATHMON Environment		Perform System Shutdown

VST030.vsd

The following briefly describes the tasks shown in [Figure 1-7](#) on page 1-20:

1. Start the Pathway environment.

This task consists of starting the PATHMON and PATHCOM processes. The PATHCOM process provides the interactive command interface to the PATHMON process.

2. Configure and start the PATHMON environment.

- Configure the PATHMON process. You can specify a backup processor and request error dumping.
- Specify global limits and settings with the SET PATHWAY command. Use the cold start option to start the environment for the first time. Use the cool start option to start the environment using an existing configuration.
- Start the PATHMON environment by using the START PATHWAY command.

3. Configure the PATHMON-controlled objects: that is, the set of objects controlled by the PATHMON process. Note that this step is optional if you cool started the PATHMON environment in Step 2, that is, if you configured these objects in a previous session and do not need to configure any of them.

- Define configuration parameters for terminal control processes (TCPs) with the SET TCP command and then add the TCPs to your PATHMON environment with the ADD TCP command.
- Define configuration parameters for terminals and other input-output devices with the SET TERM command and then add the devices with the ADD TERM command.
- Define configuration parameters for SCREEN COBOL program templates with the SET PROGRAM command and then add the program templates with the ADD PROGRAM command.
- Define configuration parameters for server classes with the SET SERVER command and then add the server classes with the ADD SERVER command.

4. Start the PATHMON-controlled objects.

- Start all the TCP, TERM, and SERVER objects by using the appropriate START command. For example, the command START SERVER * starts all static server processes defined for all server classes.
- Run process threads based on your program templates by using the RUN PROGRAM command.

5. Maintain your PATHMON environment by performing these tasks as needed:

- Monitor the status and performance of PATHMON-controlled objects.
- Monitor exception conditions.
- Send messages to terminal users.

- Reconfigure objects to correct problems and improve performance.
6. Stop your PATHMON environment.

Use the SHUTDOWN2 command to stop your system in an orderly way so you can restart easily later. (Although the SHUTDOWN command is supported for backward compatibility, the SHUTDOWN2 command is recommended.)

The SHUTDOWN2 command also provides options for faster shutdown in emergencies.

The tasks in [Figure 1-7](#) on page 1-20 shows only PATHMON-controlled objects—described in detail throughout the remaining sections of this manual. The tasks not specific to Pathway/iTS —such as starting the PATHMON environment and managing other objects—are described in detail in the *TS/MP System Management Manual*.

For complete information about the PATHCOM commands that operate on Pathway/iTS objects, including syntax, usage rules, and error messages, see [Section 7, Overview of PATHCOM](#), through [Section 12, Tell Message Commands](#). For a summary of these commands, see [Appendix A, Syntax Summary](#). For complete information about the PATHCOM commands that operate on TS/MP objects, see the *TS/MP System Management Manual*.

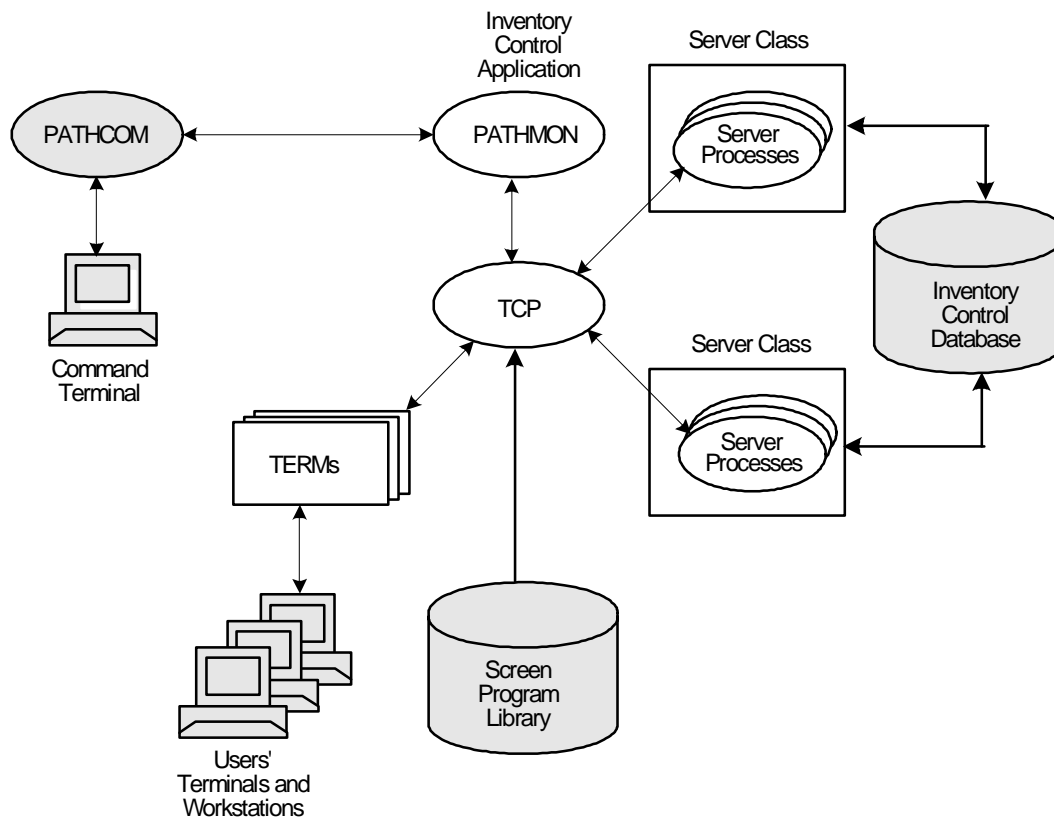
Configuring Pathway/iTS Objects

The tools to configure and manage a PATHMON environment, including the PATHMON process and the PATHCOM interface, are provided by TS/MP. Comprehensive instructions for configuring and managing a PATHMON environment, in addition to the SERVER objects in that environment, are given in the *TS/MP System Management Manual*. This overview provides a look at how objects created under the HP NonStop Pathway/iTS product—TCPs, TERM objects, and PROGRAM objects—are configured in relation to other objects and processes in a PATHMON environment.

To configure a PATHMON environment—that is, the set of objects controlled by a PATHMON process—you specify global limits that define the number of such objects in the environment, and you configure the TCPs, TERM objects, PROGRAM objects, and SERVER objects that run under the PATHMON process to support your application.

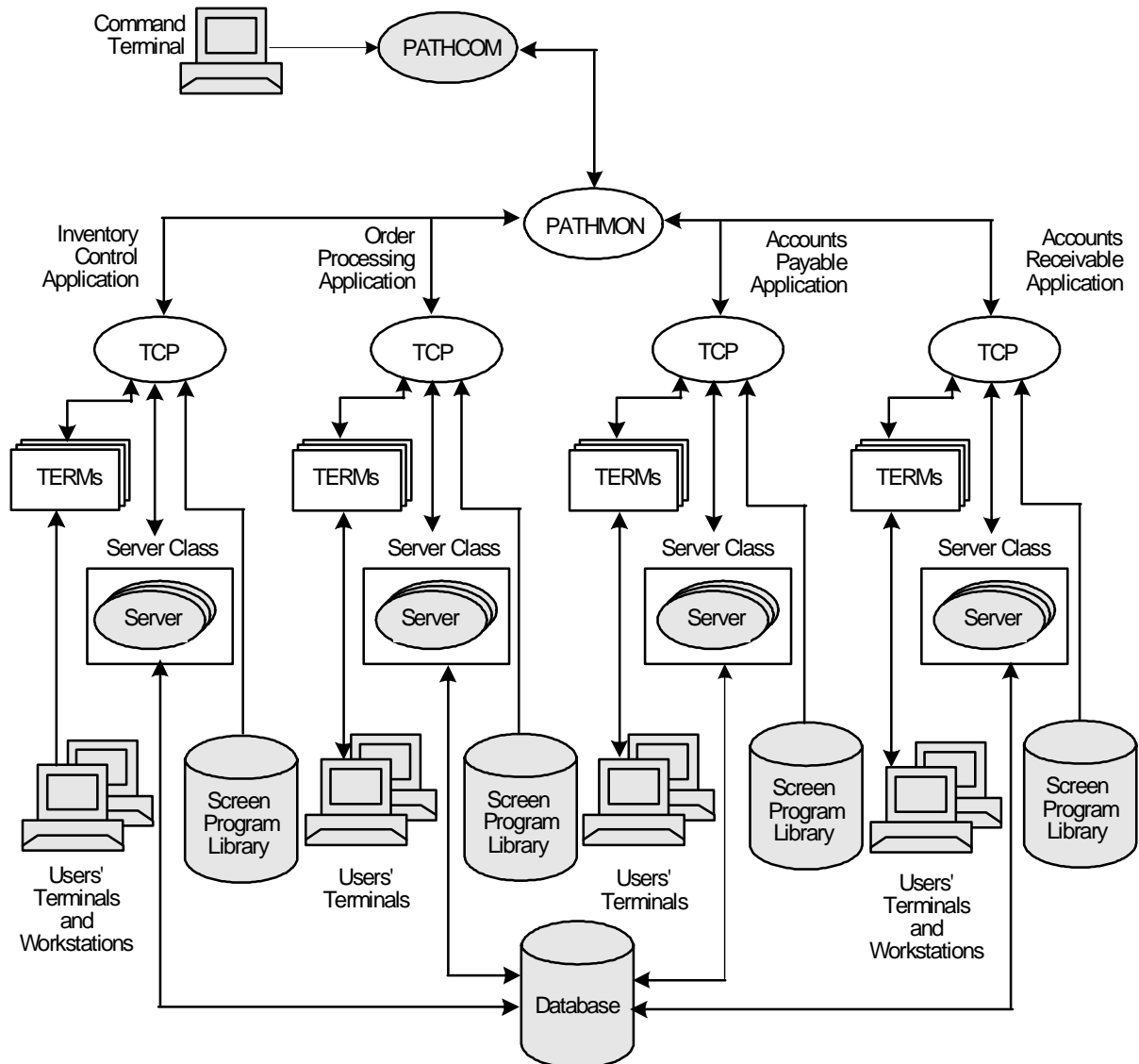
Configuration Overview

A PATHMON environment can include one application or multiple applications. For example, a PATHMON environment might consist of only one application, devoted to inventory control, as shown in [Figure 2-1](#) on page 2-2.

Figure 2-1. PATHMON Environment With a Single Application

VST021.vsd

A PATHMON environment might also consist of multiple applications such as order processing, accounts payable, accounts receivable, in addition to inventory control, as shown in [Figure 2-2](#) on page 2-3. Some applications might use terminals only, some workstations (running browser-based clients) only, and some a combination of the two.

Figure 2-2. PATHMON Environment With Multiple Applications

VST008.vsd

In [Figure 2-2](#), each application is configured under its own TCP and all access the same database. Alternatively, all applications could be configured under the same TCP, with each application accessed through a main menu. In addition, each application might have its own database.

In some configurations, a single application spans multiple PATHMON environments. This kind of configuration is often used in support of Pathway applications distributed over a network. In cases where more than one PATHMON environment is involved, each environment is monitored by its own PATHMON process.

How you configure a PATHMON environment depends on various management objectives: the size of your application or applications, the number of users, response-time requirements, whether the application is distributed over a network, and so on.

For example, a very large application—such as one supporting 100 terminal control processes, 1000 server classes, 40,000 links and 2000 terminals—might be configured in a single PATHMON environment with several TCPs to handle the large number of TERM objects required for the application. If you are running such a system on a single node, you might, for performance reasons, choose to create a separate PATHMON environment to manage the server classes and their associated links. By distributing the application among multiple processes, you improve response time.

Configuring Limits for Pathway/iTS Objects

The SET PATHWAY command requires you to define limits for TCP, TERM and PROGRAM objects configured under the Pathway/iTS product. Limits are global values that determine the maximum number of objects of each type you can define when you are ready to configure individual objects. For example, suppose that you want to indicate that your system will permit a maximum of 2 TCPs and 50 TERM objects. To establish these limits, you enter these commands:

```
= SET PATHWAY MAXTCPS 2  
= SET PATHWAY MAXTERMS 50
```

For more information about use of the SET PATHWAY command to configure these Pathway/iTS limits, including syntax descriptions, see [Section 8, Pathway Environment Control Commands](#). For additional information on configuring, starting, and managing your overall PATHMON and Pathway environments, see the *TS/MP System Management Manual*.

Specifying Limits

You must use the SET PATHWAY command to define these required limits before executing a START PATHWAY command:

- MAXTCPS—specifies the maximum number of TCP descriptions that you can add to the PATHMON configuration file.
- MAXTERMS—specifies the maximum number of descriptions for dynamic terminals, static terminals, or dynamic and static terminals that you add to the PATHMON configuration file using the ADD TERM and RUN PROGRAM commands.

In addition to the required limits, you can specify limits for other attributes. For other attributes, if you do not specify a limit, a default is assigned. For example, MAXPROGRAMS is the maximum number of PROGRAM descriptions that you can add to the PATHMON configuration file.

Specifying limits always involves some guessing, with penalties if you estimate wrong. Begin with a thorough understanding of your business application environment. Then

choose your parameters very carefully. Once you start your PATHMON environment, you cannot alter these limits without shutting it down for complete reconfiguration.

It is recommended that you always allow some space for growth. For example, if you are not certain whether you might need an extra TCP, leave room for one or two. Providing for a few more objects than you initially need can save you much unnecessary work later. If you specify unreasonably large limits, however, you cause the PATHMON process to allocate unused virtual storage and its corresponding swap-file space.

For more information about limits and default values, see [Appendix C, Configuration Limits and Defaults](#). Note that identifying a truly optimum configuration for objects configured under the Pathway/iTS product can require a significant degree of calculation and tuning that is beyond the scope of this manual. For additional help, contact your service provider for details about performance classes offered and for a description of services available.

Specifying Node Independence

Using the NODEINDEPENDENT attribute of the SET PATHWAY command makes unspecified node names for processes and devices (for both TS/MP and Pathway/iTS objects) default to the node where the PATHMON process is running. Using this attribute in your PATHMON configuration facilitates switching your Pathway application to another node, either in the event of failure or as part of a planned migration (for example, in a configuration that includes Nomadic Disk technology). When the PATHMON process is cool started on the new node, the unspecified node names are replaced by the name of that node, thereby migrating the associated objects.

If you do not specify a node name as part of the file name for a process or device in a configuration parameter, the default node for file-name expansion is affected by any values you specify for the TS/MP CMDVOL command or for the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. The node name used is the node on which the PATHMON process is currently running if any one of these situations applies:

- You set the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command to ON and you omit the node name in the parameter
- You set the node name to * in a CMDVOL command and you omit the node name in the parameter
- You specify * for the node name in the parameter

For further general information about node independence, see the *TS/MP System Management Manual*. For specific information about migrating Pathway/iTS device names to a new node, see [Section 4, Maintaining Pathway/iTS Objects](#).

Configuring TCP, TERM, and PROGRAM Objects

After specifying global limits and issuing the START command (as described in the *TS/MP System Management Manual*), you configure TCPs and the TERM, PROGRAM and SERVER objects that run under the PATHMON process to support your application. Note that only the TCPs and the TERM and PROGRAM objects are configured as part of the Pathway/iTS product. SERVER objects are configured with TS/MP, described in the *TS/MP System Management Manual*.

You create and control objects using PATHCOM commands and naming conventions that you determine. The SET and ADD commands define and change object attributes. The SHOW command allows you to display attributes for an object. PATHCOM maintains a list of attributes that describe configuration information for each object: how it relates to other objects and how it should be managed by the PATHMON process. The attributes for a TCP, for instance, specify the TCP's name, the processors on which the TCP runs, the TCP's execution priority, the name of the file that contains the object code that the TCP runs, and other characteristics of the TCP.

For a complete description of the SET, ADD, and SHOW commands and guidelines for using them, in addition to guidelines for object naming conventions, see the *TS/MP System Management Manual*. For descriptions of the SET, ADD, and SHOW commands as they apply to TCP, TERM, and PROGRAM objects, see Sections 9, 10, and 11, respectively, in this manual.

Configuring TCPs

As a multitasking process, a TCP can concurrently manage many user terminals and execute many SCREEN COBOL programs. To accomplish this work, the TCP runs its TERM objects as tasks, interleaving the concurrent processing of requests from many input-output devices or processes. This multithreading allows the TCP to handle complex groups of operations for many users at the same time.

The tasks performed by the TCP include:

- Interpreting and executing a SCREEN COBOL program for each TERM or PROGRAM object (for web clients, this program is the gateway program provided by Pathway/iTS)
- Verifying and storing terminal context data in an extended data segment
- Establishing links for screen programs with server processes
- Coordinating with the router processes for web clients and sockets IDS requesters
- Servicing SCREEN COBOL SEND requests by sending request messages to server processes and receiving reply messages back
- Gathering statistics about TERM objects, server processes, and the TCP itself

- Reporting operating errors and status information to the PATHMON process

For Pathway/iTS web clients, the gateway program interpreted and executed by the TCP validates user access and performs functions such as beginning and ending transactions, user conversion routines, unsolicited message processing, and send operations to Pathway server processes.

Defining Attributes

You use the SET TCP and ADD TCP commands to define the attributes for a TCP, name it, and add it to the system. For all TCPs, the PATHMON process requires you to specify these attributes:

- MAXTERMS specifies the maximum number of configured and temporary TERM objects that the TCP can have open at the same time.
- TCLPROG specifies the object library file that the TCP searches to locate the screen programs for its TERM objects. If the TCP supports web clients, this object file must include the GWY object program provided as part of Pathway/iTS.

In addition to specifying the required attributes, you can either specify or accept default values for other, optional attributes. Examples of these optional attributes are:

- PROCESS specifies the process name of the TCP.
- MAXSERVERCLASSES specifies the maximum number of server classes with which the TCP can communicate simultaneously.
- MAXSERVERPROCESSES specifies the maximum number of server processes, distributed among all server classes, with which the TCP can communicate simultaneously.
- CODEAREALEN specifies the number of bytes that the TCP allocates in its extended data segment for SCREEN COBOL object code.
- SERVERPOOL specifies the number of bytes that the TCP allocates for I/O requests and replies between SCREEN COBOL programs and server processes.
- TERMPOOL specifies the number of bytes that the TCP allocates in its data area for all terminal I/O buffers.
- TERMBUF specifies the maximum number of bytes that the TCP allocates from the TERMPOOL area for its terminal output buffers.
- MAXREPLY specifies the maximum number of bytes permitted for an outgoing SEND message or a server reply message.
- MAXTERMDATA specifies the number of bytes that the TCP allocates for context data for each terminal. The TCP uses this number and the value of MAXREPLY to allocate the size of the context areas (Slot 0 and Slot 1) for its terminals.
- HIGHPIN specifies whether the TCP runs at a high PIN or a low PIN.
- CPUS specifies the primary and backup processors in which the TCPs run.

- PRI specifies the priority at which TCPs run. You should coordinate with the system manager for your PATHMON environment when setting this attribute, to ensure that TCP priorities are compatible with priorities set for server classes.

In general, priorities should facilitate the completion of work over the arrival of work. Usually, this means that server classes should have a higher priority than TCP objects. If there are multiple server classes in a transaction path, however, and the SCREEN COBOL code is simple, continuing a transaction with multiple server classes might mean waiting for TCP availability.

Note. If you are configuring a TCP for use with Pathway/iTS web clients, the gateway requester program requires that you configure specific values for certain TCP attributes, as follows:

```
SERVERPOOL 40000
TERMPPOOL 200000
TERMBUF 32000
MAXREPLY 32000
MAXTERMDATA 200000
```

Otherwise, gateway TERM threads may be suspended at run time.

For a detailed description of all the SET TCP attributes, see [Section 9, Terminal Control Process \(TCP\) Commands](#).

For example, suppose that you want to define and add a TCP named TCP-PRIME, whose primary process runs in CPU 1 and whose backup process runs in CPU 0. The TCP runs at a high PIN. To locate its screen programs, the TCP must search a directory named \$DATA.APPLIC.POBJ. This TCP can open maximums of 50 TERM objects, 5 server classes, and 25 server processes across the 5 server classes.

You can define and add this TCP by entering these commands:

```
= SET TCP CPUS 1:0
= SET TCP HIGHPIN ON
= SET TCP MAXTERMS 50
= SET TCP TCLPROG $DATA.APPLIC.POBJ
= SET TCP MAXSERVERCLASSES 5
= SET TCP MAXSERVERPROCESSES 25
= ADD TCP TCP-PRIME
```

As another example, suppose that you want to define and add a TCP named TCP-WEB, whose primary process runs in CPU 5 and whose backup process runs in CPU 4. To locate its screen programs, the TCP must search a directory named \$DATA.WEBCONN.GWAY, which contains the GWAY program provided as part of Pathway/iTS. This TCP can open a maximum of 100 TERM objects.

```
= SET TCP CPUS 5:4
= SET TCP MAXTERMS 100
= SET TCP TCLPROG $DATA.WEBCONN.GWAY
= SET TCP SERVERPOOL 40000
= SET TCP TERMPPOOL 200000
= SET TCP TERMBUF 32000
```

```
= SET TCP MAXREPLY 32000  
= SET TCP MAXTERMDATA 200000  
= ADD TCP TCP-WEB
```

Distributing the Transaction Load Across TCPs

In general, you want to spread the workload across all available resources. In this case, the load is the transaction load generated by web clients, terminals, or intelligent devices.

For web clients and sockets IDS requesters, the router processes distribute connection requests to started TERM objects in such a way that the workload is optimally balanced among all the TCPs in the Pathway environment. However, for other requesters, you should configure your TCPs for load balancing.

Assuming that all terminal activities generate the same load, each TCP should have the same number of terminals configured to it. (The SET TCP MAXTERMS command specifies the number of terminals that a TCP can have open at the same time.)

You will see a more consistent response time if the load is evenly distributed across TCPs; however, this might work against manageability. In general, when processor utilization for the TCPs is comparable, you can assume that the terminal load is evenly spread across them.

Requesting Error Dumping

You can request error dumping by specifying either the SET TCP, DUMP ON command or the CONTROL TCP, DUMP ON command.

- Use the SET TCP command if you want to request error dumping and you have not yet started the TCP.
- Use the CONTROL TCP command if you want to request error dumping but you have already started the TCP. For more information about using this command, see [Information to Include When Reporting Problems](#) on page 4-21.

In case of an internal or fatal error, the TCP generates an error dump and writes it to a file. For example, this command directs the TCP to write error information to a file named TCPDUMP:

```
= SET TCP, DUMP ON (FILE TCPDUMP)
```

After this command is entered and the TCP is started, if an internal or fatal TCP error occurs, the TCP writes the contents of its data stack and its extended data segment to TCPDUMP.

If you request the help of your service provider in analyzing a problem, the representative will likely require a DUMP file. It is therefore recommended that you always set the DUMP option to ON for production systems.

Configuring TCPs for a Customized TCP Object Library

You are allowed nine user conversion procedures for data passed between a SCREEN COBOL program and a terminal screen or intelligent device. These procedures are stored in the TCP user library object file, `$SYSTEM.SYSTEM.PATHTCPL`. For web clients converted from SCREEN COBOL programs, these user conversion procedures are run by the Pathway/iTS gateway process.

You can write your own user conversion procedures to replace those provided in the standard TCP user library object file, and then customize the library object file to include these procedures by using the `nld` program. A macro called `MAKEUL` simplifies the task of building the customized object file. The *HP NonStop Pathway/iTS TCP and Terminal Programming Guide* describes these tasks in detail.

You have two choices as to how you name the library object file:

- You can name the customized library `$SYSTEM.SYSTEM.PATHTCPL`.
- You can name the customized library file something other than `$SYSTEM.SYSTEM.PATHTCPL`, such as `$SKY.KING.USERTCPL`.

If you keep the name `$SYSTEM.SYSTEM.PATHTCPL` for your customized library file, you need not explicitly configure the library file name. If you use a different name, you must use a `SET TCP` command with the `GUARDIAN-LIB` option to name the library file:

```
= SET TCP GUARDIAN-LIB $SKY.KING.USERTCPL
```

To ensure that the proper user library object file is always used, use the `SET TCP` command with the `GUARDIAN-LIB` option as part of your standard `PATHMON` environment startup procedure,

△ **Caution.** Each time you install a new version of the TCP, you should add your customized procedures to the new `PATHTCPL` object library file (be sure you name the new library file the same name as the former, customized library file). Using the latest `PATHTCPL` object library combined with your procedures ensures that there are no incompatibilities between the new version of the TCP and the `PATHTCPL` object library.

Configuring TERM and PROGRAM Objects

Terminal objects represent tasks controlling the input-output devices and processes that allow users to interact directly or programmatically with a Pathway application. There are two types of these objects: configured terminal (TERM) objects and temporary terminal (PROGRAM) objects.

Configured TERM objects are those you define and add to the `PATHMON` configuration file. Temporary TERM objects are created for you (and automatically deleted) by the `PATHMON` process in response to a `RUN PROGRAM` command, using a template you configure as a PROGRAM object. The names you specify for configured TERM objects must begin with a letter; the `PATHMON` process assigns names beginning with numbers to temporary TERM objects.

For Pathway/iTS web clients and sockets IDS requesters, you must use configured TERM objects. However, for other intelligent devices and for terminals, temporary TERM objects can be easier to use than configured TERM objects. Because more default attributes are supplied for a PROGRAM object than for a TERM object, it is easier to use a PROGRAM object. And because temporary TERM objects are deleted automatically when the user completes a session and exits the program, PROGRAM objects can be easier to manage from an operations standpoint.

On the other hand, a Pathway application can support just 100 concurrent waited RUN PROGRAM requests from PATHCOM processes and SPI processes. Thus, it may be appropriate to use temporary TERM objects to control input-output devices and processes only in a low-volume, low-terminal-count, multiple-application environment. An appropriate environment, for example, might be a system management area in which 10 to 15 operator terminals monitor the system using several different information-gathering Pathway applications.

A Pathway application can support up to 4095 configured TERM objects. Thus, it is often appropriate to use TERM objects for large networks of terminals dedicated to a single application. A high-volume application that includes many non-dedicated devices must be coded to dynamically add and delete TERM objects as needed when the non-dedicated devices access the application.

Configuring TERM Objects

For each input-output device or process you want to configure explicitly, you must define and add a TERM object using the SET and ADD commands. A TERM object runs as a thread in the TCP, which can handle many such TERM objects concurrently.

For example, consider a PATHMON environment that runs a single order-processing application on 100 application workstations or terminals. Whenever users begin to use the application, their web browsers or terminals typically display the logon or menu page or screen for the application, as presented by the initial screen program. The users can use only this application; they cannot access other software. To support this system, you would need to configure 100 TERM objects—one for each workstation or terminal.

To configure a TERM object, you must specify these attributes with the SET TERM command:

- **FILE** specifies the name of the input-output device or process that the TERM controls.

This can be the name recognized by TACL for a terminal or other physical device. Alternatively, it can be the name of a process that handles work such as front-end processing or terminal emulation. For TERM objects supporting web clients and sockets IDS requesters, it must be the name of the associated router process.

- **INITIAL** specifies the name of the first screen program that runs on the device or process when the TERM object is started. For TERM objects supporting web

clients, this must be GATEWAY, the name of the initial screen program in the gateway requester.

This is the name specified in the PROGRAM-ID sentence of the Identification Division in a SCREEN COBOL program.

- TCP specifies the name of the TCP that controls the screen program and oversees communication between the screen program and the input-output device or process.

In addition to the required attributes, other, optional ones can be selected:

- TYPE specifies the type of device supported by the TERM object. For TERM objects supporting web clients, TYPE is required and must be specified as INTELLIGENT.
- BREAK specifies whether the TCP accepts the Break key function for its terminals running in conversational mode.
- DIAGNOSTIC specifies whether diagnostic screens are displayed to inform the terminal operator when an error condition or termination occurs.

For a complete description of all the SET TERM attributes, see [Section 10, TERM Commands](#).

Suppose that you suppose that you want to add two physical terminals, with device names \$JOAN and \$SAM, and you want to control them through TERM objects named TERM-007 and TERM-008, respectively. These TERM objects will run as threads in TCP named TCP-PRIME, and upon activation, enter a screen program named MAIN-MENU. The terminals are 6530 terminals. You can define and add these TERM objects by entering:

```
= SET TERM INITIAL MAINMENU
= SET TERM TCP TCP-PRIME
= SET TERM TYPE T16-6530:0
= ADD TERM TERM-007, FILE $JOAN
= ADD TERM TERM-008, FILE $SAM
```

As another example, suppose that you want to add a workstation running a browser-based client, control it through a TERM object named CLIENT-001, and associate it with the router process \$ROUT1. You can define and add the TERM object as follows:

```
= SET TERM INITIAL GATEWAY
= SET TERM TCP TCP-WEB
= SET TERM TYPE INTELLIGENT
= ADD TERM CLIENT-001, FILE $ROUT1
= ADD TERM TERM-008, FILE $SAM
```

Configuring PROGRAM Objects

PROGRAM objects are templates used by the PATHMON process when creating and starting temporary TERM objects. The templates define tasks that enable TCPs to run

screen programs temporarily on one or more devices. When the task is completed, the TERM object is automatically deleted.

When a user issues a RUN PROGRAM command, the associated PROGRAM object implicitly creates a TERM object for that individual device or process. This TERM object manages that device or process, and enables the user to interact with the Pathway application associated with the screen program. This TERM object, however, is only temporary; it ceases to exist when the user exits from the screen program.

To configure a PROGRAM object, you must specify these attributes with the SET PROGRAM command:

- TCP specifies the name of the TCP that controls the screen program.
- TYPE specifies the type of input-output device or process that the PROGRAM object controls and the name of the screen program that runs on the device or process when it is started.

You must specify one TYPE attribute for each type of device or process supported by the PROGRAM object. Then, for each type, you indicate the name of the screen program that should be executed if the PROGRAM object is run on a device of that type. (In a SCREEN COBOL program, the program name is the name specified in the PROGRAM-ID paragraph of the Identification Division.)

Because your screen programs can run on many different types of devices or processes, you can also use the TYPE attribute to specify the different options that these devices or processes require.

In addition to the required attributes, other, optional ones can be selected:

- SECURITY specifies which users are allowed to run the screen program.
- ERROR-ABORT specifies whether the TCP should abort the screen program task in the event of an error.
- BREAK, an option of the TYPE attribute, specifies whether the TCP accepts the Break key function for its PROGRAM objects running in conversational mode.

For a complete description of all the SET PROGRAM attributes, see [Section 11, PROGRAM Commands](#).

The following example defines and adds a PROGRAM object named SALES. This PROGRAM object enables a TCP named TCP-PRIME to run a screen program named MENU-6530 on 6530 terminals and a screen program named MENU-3270 on IBM 3270 terminals:

```
= SET PROGRAM TCP TCP-PRIME
= SET PROGRAM TYPE T16-6530 (INITIAL MENU-6530)
= SET PROGRAM TYPE IBM-3270 (INITIAL MENU-3270)
= ADD PROGRAM SALES
```

Configuring a Template for Several Device Types

A PROGRAM object can serve as a template for several device types. The template can optionally associate several device-related attributes with each device type. These attributes include the initial screen program for that device type, the printers associated with the device type, and the ability of the device type to accept the BREAK key function and to echo input characters on the screen. This template can be used over and over again to start many devices of the specified type or types. Through the PROGRAM, any device of the appropriate type can access the application.

Specifying Security for a PROGRAM Object

You can specify security for a PROGRAM object by setting the OWNER and SECURITY attributes of the SET PROGRAM command. The OWNER attribute specifies the owner of the current PROGRAM object. The owner can alter the security attribute for the PROGRAM object. The SECURITY attribute (A, G, O, -, N, C, or U) values are the same as those for *Guardian* operating environment security attributes. The SECURITY attribute specifies who can run the PROGRAM object.

The following example specifies that only the owner—user ID 8,61—can run this PROGRAM object:

```
= SET PROGRAM OWNER 8,61  
= SET PROGRAM SECURITY "O"
```

The next example specifies that any local or remote user can run the PROGRAM object:

```
= SET PROGRAM OWNER 8,61  
= SET PROGRAM SECURITY "N"
```

Communication Between PATHMON Environments

When one PATHMON environment communicates with another, the PATHMON process for each environment controls and reports on its own objects. When a TCP in one PATHMON environment executes a screen program request directed to a server class controlled by another PATHMON process, these operations occur:

1. The TCP requests a link from the external PATHMON process that controls the server class.
2. The external PATHMON process grants the link to the server process.
3. The TCP uses the link to open the remote server process and send the request to that server process.
4. After the remote server process completes its work, it returns its reply to the TCP.

The same general operations take place whether the PATHMON environments are running on the same or on different nodes.

For communication between PATHMON environments, the TCP requesting a server link must pass all process security checks. If the TCP and server class are on different nodes, the security requirements demand the following:

- The person who starts the local PATHMON process must possess a remote password for the external PATHMON process's node.
- The person who starts the external PATHMON process must possess a remote password for the local PATHMON process's node.

This level of security is required because the local TCP must be able to open the external PATHMON process and server classes, and the external PATHMON process must be able to open the local TCP. All TCPs and server processes started by a PATHMON process run under the user ID of the person who started that PATHMON process.

Starting and Stopping Pathway/iTS Objects

This section describes how to start and stop HP NonStop Pathway/iTS objects, including the router process and the PATHMON-controlled objects—TCPs, TERM object, and PROGRAM objects.

Starting Router Processes

Pathway/iTS router processes are not part of the PATHMON environment; you must start them in the *Guardian* operating environment. If you are using Pathway/iTS web clients or sockets IDS requesters, you must start one or more router processes before you start the PATHMON environment.

You start a router process as a named process from the TACL prompt as follows:

```
RUN ROUTER / NAME $name , NOWAIT , CPU cpu / port protocol  
[ tcp-ip ] [ backup-cpu ]
```

\$name

is the name to be assigned to the router process.

cpu

is the primary processor in which the process is to run.

port

is the TCP/IP port number to be assigned to the router process.

protocol

is the protocol type to be used by the router process (HTTP or SOCKET).

tcp-ip

is the name of a TCP/IP process running on the HP NonStop system. If omitted, the default TCP/IP process is used (\$ZTC0 or the process named in the DEFINE for TCPIP^PROCESS^NAME), as described in the *TCP/IP and IPX/SPX Programming Manual*.

backup-cpu

is the backup processor. If omitted, the router process runs without a backup.

For example:

```
TACL > RUN ROUTER /NAME $ROUT1, CPU 5, NOWAIT/ 8857 HTTP $ZTC1 4
```

If the specified TCP/IP process does not exist, or if the specified TCP/IP port number is a reserved port or is being used by another process, the router process reports an error.

You can ensure connection availability and reliability by running the both the router process and the TCP as process pairs. To do this, configure the CPUS attribute of the TCP to designate both a primary and a backup process, as described in [Configuring TCPs](#) on page 2-6, and specify the *backup-cpu* parameter when you run the router process. If the primary router process fails, the backup router takes over and maintains the queued client connection requests and the TERM objects' requests for connections. It also maintains the currently established load balancing.

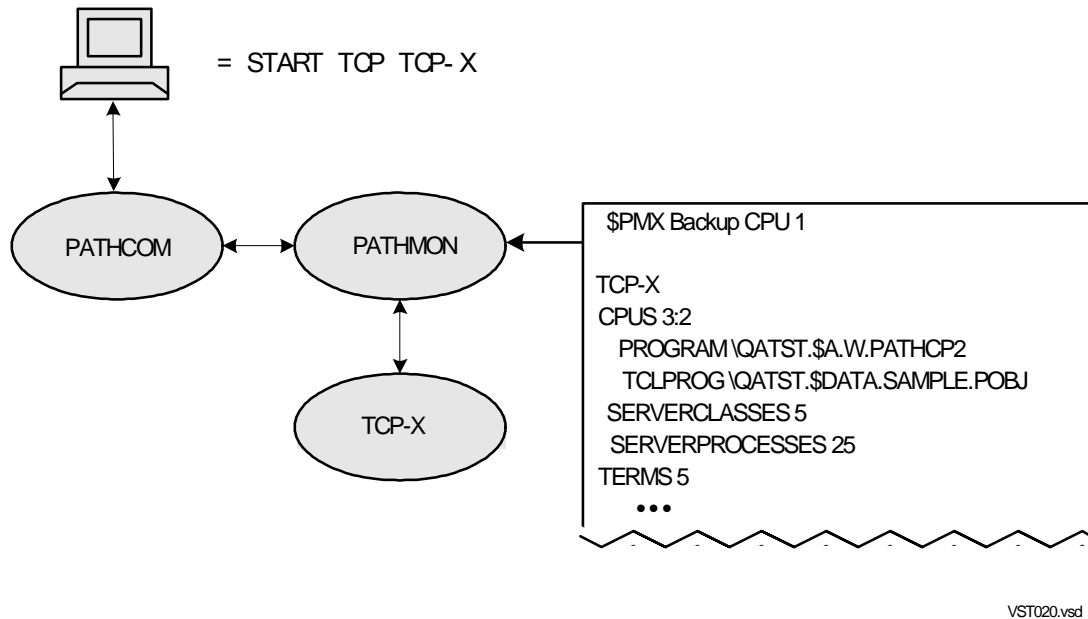
Starting PATHMON-Controlled Objects

After you configure and start your PATHMON environment and define and add each TCP, TERM, PROGRAM, and SERVER object, you issue the START command to activate each object or process. (Recall that the PATHMON environment and its SERVER objects are defined and managed under TS/MP, as described in the *TS/MP System Management Manual*.)

For example, this command starts a TCP named TCP-X:

```
= START TCP TCP-X
```

When the PATHMON process executes the START command, it checks the status of the object named and then performs the operations needed to start the object, as shown in [Figure 3-1](#) on page 3-3.

Figure 3-1. Starting a PATHMON-Controlled Object

You must start some objects before starting others. For example, you must start a TCP before starting the TERM objects that use the TCP. (Until a TCP is running, neither device operation nor screen program execution can begin.)

Once you start the TCP, you can start the terminal objects under that TCP's control. For example, this command starts the TERM object NATHAN:

```
= START TERM NATHAN
```

You can start a static server process before or after you start TCP and TERM objects. Server process creation does not depend on the status of the TCP or TERM objects in a PATHMON environment.

You can expect faster initial response time, however, if you start server processes before you start TCP and TERM objects. You can start a static server process anytime after you configure the server class to which it belongs.

Note. Server processes are part of TS/MP. For information on managing server processes, see the *TS/MP System Management Manual*.

When starting multiple objects of the same type, you can list specific object names in a single START command. The following command, for example, starts three TERM objects:

```
= START TERM (NATHAN, ERIC, KEVIN)
```

Alternatively, you can start all PATHMON-controlled objects of the same type by using the asterisk option, as shown in the next command:

```
= START TERM *
```

Because the asterisk option slows processing, a more efficient way to start all objects is to use an command file in which all the objects are named. Use the asterisk option if no command file is available.

If an object is already running when you issue the START command, the PATHMON process displays an error message and makes no attempt to start the object.

Starting TCPs

To start a TCP process, a set of TCPs, or all TCPs under the control of a given PATHMON process, use the START TCP command.

For example, the following command starts the TCP process named TCP-PRIME:

```
= START TCP TCP-PRIME
```

The next command starts the TCPs named TCP-1 and TCP-2:

```
= START TCP (TCP-1, TCP-2)
```

The next command starts all defined TCPs:

```
= START TCP *
```

To start a TCP, the PATHMON process creates the TCP process by running the program file indicated by the PROGRAM attribute of the TCP object, as recorded in the PATHMON configuration file. This attribute specifies the name of the file that contains the TCP's object code. The PATHMON process then passes the TCP definition to the TCP process. The TCP uses the information in the TCP definition to initialize itself and start its backup process in the designated CPU.

Starting TERM Objects

To start a TERM object, a set of TERM objects, or all TERM objects under the control of a given PATHMON process, use the START TERM command.

For example, the following command starts the TERM named TERM-001:

```
= START TERM TERM-001
```

The next command starts the TERM objects named TERM-002 and TERM-003:

```
= START TERM (TERM-002, TERM-003)
```

The next command starts all TERM objects under the control of TCP-1 that are not running:

```
= START TERM *, TCP TCP-1, STATE NOT RUNNING
```

The next command starts all TERM objects defined in the PATHMON configuration file:

```
= START TERM *
```

After reading the TERM definition from the PATHMON configuration file, the PATHMON process passes the TERM definition to the TCP process specified by the TCP attribute in this definition. When the TCP receives this information, it:

- Checks the specified library file for the program indicated by the INITIAL attribute.
- Opens communication with the terminal, device, or process specified by the FILE attribute.
- Executes the screen program code.

If you enter a START TERM command before the TCP that controls the TERM is running, the PATHMON process displays an error message.

Starting Multiple TCPs and TERM Objects in Parallel

When the PATHMON process has configured many TCP and TERM objects, they can take a long time to start because the TERM objects are started one at a time for each TCP. After all the TERM objects for the first TCP are started, the TERM objects for the next TCP in the configuration are started again, one by one and so on, until all the terminals for all TCPs are started.

You can speed this process by using multiple PATHCOM processes to start your TCP and TERM objects in parallel. (In such a case, it is convenient to enter your PATHCOM commands through a command file executed by an OBEY command.)

To start the TCPs and TERM objects as quickly as possible without using an command file, you can start a PATHCOM process for each TCP configured (be sure to use the NOWAIT option). In your PATHCOM command, use the START TCP command and START TERM command with the TCP option, as shown in this example. (It is faster to log output to the spooler than to a terminal or printer.)

```
4> PATHCOM /OUT $$, NOWAIT/ $PMX; ERRORS 99; START TCP1;
    START TERM *, TCP TCP1, SEL NOT RUNNING
5> PATHCOM /OUT $$, NOWAIT/ $PMX; ERRORS 99; START TCP2;
    START TERM *, TCP TCP2, SEL NOT RUNNING
6> PATHCOM /OUT $$, NOWAIT/ $PMX; ERRORS 99; START TCP3;
    START TERM *, TCP TCP3, SEL NOT RUNNING
7> PATHCOM /OUT $$, NOWAIT/ $PMX; ERRORS 99; START TCP4;
    START TERM *, TCP TCP4, SEL NOT RUNNING
8> PATHCOM /OUT $$, NOWAIT/ $PMX; ERRORS 99; START TCP5;
    START TERM *, TCP TCP5, SEL NOT RUNNING
9> PATHCOM /OUT $$, NOWAIT/ $PMX; ERRORS 99; START TCP6;
    START TERM *, TCP TCP6, SEL NOT RUNNING
```

(The number of PATHCOM processes you can start is defined by the SET PATHWAY MAXPATHCOMS command. For information on setting global limits for the PATHMON environment, see the *TS/MP System Management Manual*.)

Starting PROGRAM Objects

To start a temporary terminal (PROGRAM) object, use the RUN PROGRAM command. For a discussion of temporary and configured terminal objects, see [Section 2, Configuring Pathway/iTS Objects](#).

For example, the following command, entered at an application terminal, runs the application named PROG-1:

```
= RUN PROGRAM PROG-1
```

In response, the PATHMON process starts the objects needed by the application as follows:

- Starts the required TCP (specified by the TCP attribute in the PROGRAM definition) if the TCP is not already running.
- Creates a TERM object for the user's terminal, based upon information from the PROGRAM definition and the RUN command.

The PATHMON process generates a TERM name for the terminal by using the system number, the terminal name, and a unique hexadecimal number that the PATHMON process supplies (for example, 077-TH0-A56). The generated name is 15 characters or fewer in length, including the hyphens.

- Starts the TERM object.
- Starts server processes as needed.

When the RUN operation terminates, the PATHMON process does the following:

- Deletes the TERM object from the system.
- Stops the TCP if the TCP was started as a result of the RUN PROGRAM and no other TERM objects were started for the TCP.
- Dissolves all links from the TCP to the server processes.

Note. You are not required to start TCP processes that are used only by users at temporary terminal devices: the PATHMON process starts these objects as needed. Your application has better response time, however, if you start objects before they are actually required. Note that the PATHMON process is provided as part of TS/MP. For more information about the PATHMON process and its functions, see the *TS/MP System Management Manual*.

You can run a PROGRAM object with or without the NOWAIT option.

The NOWAIT option enables a maximum of 4,095 PROGRAM objects to run at the same time. If you do not specify NOWAIT, the maximum number of PROGRAM objects that can run at the same time drops to approximately 100.

The NOWAIT option causes the PROGRAM object to execute concurrently with PATHCOM, so that the PATHCOM prompt returns immediately to your screen. Consequently, you must specify a terminal that is different from the terminal at which you issue the RUN PROGRAM command. The following example (entered at \$TERMA) runs the PROGRAM named SALES on terminal \$TERMB:

```
= RUN PROGRAM SALES, NOWAIT, FILE $TERMB
=
```

After you enter the RUN PROGRAM command, the PATHCOM prompt immediately appears on \$TERMA; you can now enter other PATHCOM commands while SALES runs.

If you do not specify the NOWAIT option, the PROGRAM runs at the same terminal from which you specified the RUN PROGRAM command. The PATHCOM prompt does not appear until you exit the PROGRAM:

```
= RUN PROGRAM SALES
      .
      (SALES application is running)
      .
      (user exits SALES)
=
```

Note that PATHCOM is provided as part of TS/MP. For more information on the PATHCOM process and commands, see the *TS/MP System Management Manual*.

Note. PATHCOM is provided as part of TS/MP. For more information on the PATHCOM process and commands, see the *TS/MP System Management Manual*.

Stopping Router Processes

You can stop a router process by issuing a TACL STOP command. You should usually stop router processes only after you have shut down the Pathway environment.

You stop a router process from the TACL prompt as follows:

```
STOP { $name
      { cpu, pin } }
```

\$name

is the name to be assigned to the router process.

cpu, pin

is the CPU number and process identification number of the router process.

Stopping PATHMON-Controlled Objects

You can stop a TCP or TERM object while your Pathway environment is running by issuing the STOP command.

You cannot stop a PROGRAM object; however, you can stop the temporary TERM objects associated with a PROGRAM object. (A PROGRAM object does not have an operating state with respect to the PATHMON process, and its status cannot be requested through PATHCOM. A PROGRAM object is simply a mechanism for creating temporary TERM objects as they are needed.)

Stopping TCPs

To stop a TCP, use the STOP TCP command. You can only stop a TCP after you have stopped all TERM objects controlled by that TCP.

The following command stops the TCP named TCP-EXT1:

```
= STOP TCP TCP-EXT1
```

The PATHMON process stops TCPs in the order you specify them in your STOP commands. The following command stops the TCPs named TCP-EXT2 and TCP-EXT3, in that order:

```
= STOP TCP (TCP-EXT2, TCP-EXT3)
```

The next command stops all TCPs in your PATHMON environment. The TCPs stop in alphabetical order:

```
= STOP TCP *
```

Note that the PATHMON process is provided as part of TS/MP. For more information on the PATHMON process and functions, see the *TS/MP System Management Manual*.

Note. PATHCOM is provided as part of TS/MP. For more information on the PATHCOM process and commands, see the *TS/MP System Management Manual*.

Note. The PATHCOM STOP command cannot be used to stop an external TCP, because that TCP, although linked to a local server process, is controlled by another PATHMON process. To stop an external TCP, you must issue a STOP command to the PATHMON process that controls that TCP.

Stopping TERM Objects

To stop a TERM object, use the STOP TERM command. When you issue a STOP TERM command, the PATHMON process passes your request to the TCP process controlling that TERM. The TCP waits until any critical operations underway for the TERM object are complete, and then stops the TERM object. (For example, the TCP does not stop the TERM object if a transaction protected by the TMF is in progress.)

Note that the PATHMON process is provided as part of TS/MP. For more information on the PATHMON process and functions, see the *TS/MP System Management Manual*.

This command stops all TERM objects controlled by a given PATHMON process:

```
= STOP TERM *
```

The next command stops a TERM object named TERM2:

```
= STOP TERM TERM2
```

You can also stop a TERM object by using the ABORT TERM command, as shown in this example:

```
= ABORT TERM *, TCP TCP-1
```

If a TERM object is in TMF transaction mode, the transaction is backed out before the TERM object is aborted.

Note. PATHCOM is provided as part of TS/MP. For more information on the PATHCOM process and commands, see the *TS/MP System Management Manual*.

Stopping Multiple TCPs and TERM Objects in Parallel

If you want to stop all or several TCPs and TERM objects as quickly as possible (but not shut down your PATHMON environment), you can use multiple PATHCOM processes to abort your TERM objects and stop your TCPs in parallel.

For example, you can start a PATHCOM process for each configured TCP (be sure to use the NOWAIT option) and use the ABORT TERM command and STOP TCP command, as shown. (It is faster to log output to the spooler than to a terminal or printer.)

```
4> PATHCOM /OUT $$, NOWAIT/ $PMX; ABORT TERM *, TCP TCP1;
    STOP TCP TCP1
5> PATHCOM /OUT $$, NOWAIT/ $PMX; ABORT TERM *, TCP TCP2;
    STOP TCP TCP2
6> PATHCOM /OUT $$, NOWAIT/ $PMX; ABORT TERM *, TCP TCP3;
    STOP TCP TCP3
7> PATHCOM /OUT $$, NOWAIT/ $PMX; ABORT TERM *, TCP TCP4;
    STOP TCP TCP4
```

(The number of PATHCOM processes you can start is defined by the SET PATHWAY MAXPATHCOMS command.)

Note. PATHCOM is provided as part of TS/MP. For more information on the PATHCOM process and commands, see the *TS/MP System Management Manual*.

Pathway/iTS Objects and the SHUTDOWN2 Command

You can shut down the whole PATHMON environment by stopping all PATHMON-controlled objects collectively with the SHUTDOWN2 command.

Note. This subsection describes the effect of the SHUTDOWN2 command and options on Pathway/iTS objects controlled by the PATHMON process. For a comprehensive discussion of shutting down a PATHMON environment in addition to the syntax of the SHUTDOWN2 command, see the *TS/MP System Management Manual*.

The SHUTDOWN2 command replaces the SHUTDOWN command. It is recommended that you use the SHUTDOWN2 command for a faster, more reliable means of stopping your PATHMON environment. The SHUTDOWN command, described in the *TS/MP System Management Manual*, is still available to maintain backward compatibility.

Note. Because Pathway/iTS router processes are not part of the PATHMON environment, the SHUTDOWN2 command has no effect on the router processes. You must stop these processes by using TACL.

The PATHMON process automatically stops objects in this order: TERM objects, TCP objects, SERVER objects under its control, and finally the PATHMON process itself.

When you specify a SHUTDOWN2 command, these actions occur:

- All TCP objects begin shutdown (shutting down TERM objects and then themselves) in parallel.
- New work is disallowed. For example, all ADD, ALTER, and DELETE commands are invalid. (For a complete list of commands that are disabled during shutdown, see the *TS/MP System Management Manual*.)
- The PATHMON process logs the start and completion of SHUTDOWN2; it does not log status messages during shutdown.

Three options—ORDERLY, ABORT, and IMMEDIATE—provide different levels of shutdown, as described in [Table 3-1](#) and the subsections that follow.

Table 3-1. Effects of SHUTDOWN2 Options

	ORDERLY	ABORT	IMMEDIATE
TERM objects	Stopped	Aborted	Aborted ¹
TCPs (local)	Stopped—after stopping all TERM objects under TCP's control and closing all server processes	Stopped—after aborting all TERM objects under TCP's control and closing all server processes	Stopped ²
TCPs (external)	Notified of shutdown request ³	Notified of shutdown request ³	Notified of shutdown request ³

Table 3-1. Effects of SHUTDOWN2 Options

	ORDERLY	ABORT	IMMEDIATE
Outstanding work	All outstanding work (I/O activity) is completed	For TCPs and server classes, outstanding work (I/O activity) is completed. For transactions involving an aborted TERM, unknown. ⁴	Unknown
New work	Not allowed	Not allowed	Not allowed

1. Because the TCP was stopped by a Guardian STOP procedure call, the terminals were essentially aborted. Aborting a TERM object means that transactions are not completed.
2. Indicates a potential Guardian STOP procedure call.
3. Once an external TCP (or LINKMON process) processes notification of an impending PATHMON process shutdown, that TCP or LINKMON process no longer sends data or requests to server classes under the control of the PATHMON process that is shutting down .
4. If the transaction is protected by the TMF, TMF aborts the transaction and resets any data that was changed. If the transaction is not protected by TMF, the effects of the aborted transaction are unknown.

The effects of the SHUTDOWN2 operation on a TERM object can depend on whether the STOPMODE register or TRANSMODE register is set.

If the STOPMODE register is set, the current value of the terminal's SCREEN COBOL special register is nonzero. If the TRANSMODE register is set, the terminal is currently in transaction mode. You can view the value of the STOPMODE register using the STATUS command. For more information, see [Section 4, Maintaining Pathway/ITS Objects](#).

[Table 3-2](#) describes the effects of the STOPMODE and TRANSMODE registers on the SHUTDOWN2 command.

Table 3-2. Effect of STOPMODE and TRANSMODE Registers on Shutdown Operations

	STOPMODE REGISTER SET	TRANSMODE REGISTER SET
ORDERLY	Waits for STOPMODE to be reset and waits for ACCEPT on TERM	Waits for transaction to complete and waits for ACCEPT on TERM
ABORT	Aborts TERM objects	Aborts TERM objects
IMMEDIATE	Aborts TERM objects	Aborts TERM objects

Specifying the ORDERLY Option

The ORDERLY option enables work in progress to complete before shutting down. The following command specifies an orderly shutdown:

```
= SHUTDOWN2 , MODE ORDERLY
```

The ORDERLY option requires the most time to complete the shutdown operation, because all transactions in progress are allowed to complete before objects (TCP, TERM, and server objects) involved in the transaction are stopped. For example, a TERM waiting for I/O to a server class to complete cannot be stopped until the I/O completes. (Note that the TERM object must be in a qualified state in order to be stopped. See [Table 3-2](#) and the associated discussion about the STOPMODE and TRANSMODE registers.)

You might want to selectively escalate the shutdown of some TERM objects while maintaining a generally ORDERLY shutdown. To do this, use the PATHCOM commands ABORT TERM or STOP TERM. Use the ABORT TERM command to abort a single TERM or multiple TERM objects. Use the STOP TERM command to stop a single TERM or multiple TERM objects.

Specifying the ABORT Option

The ABORT option aborts all TERMS for a faster shutdown operation. The following command specifies a shutdown in which terminals are aborted instead of stopped:

```
= SHUTDOWN2, MODE ABORT
```

The ABORT option enables sends to a server class to complete, but the TERM might be aborted before the server class can reply. Work does not necessarily stop on transaction boundaries, so the status of the transaction is unknown.

If completing outstanding transactions is important to your application, use the ORDERLY option.

Specifying the IMMEDIATE Option

To bring down the system as quickly as possible, use the IMMEDIATE option, as shown in this example:

```
= SHUTDOWN2, MODE IMMEDIATE
```

This command uses the Guardian procedure call, STOP, to stop:

- All running TCPs (that are locally controlled)
- All server processes still running after the TCP is stopped

The IMMEDIATE option is the quickest way to shut down a PATHMON environment, but you must consider these consequences before specifying this option:

- If completed successfully, the IMMEDIATE option stops objects immediately, regardless of the presence of pending requests and incomplete operations.
- During SHUTDOWN2 IMMEDIATE, the TCP is unable to reset terminals under its control. To use these terminals, you must first reset them by waking the TACL process running on the terminal, or, once the PATHMON environment is running again, start the TERM object associated with that terminal. Other TERM objects and devices might require some other form of initialization.

- The status of any outstanding transactions is unknown.

Maintaining Pathway/iTS Objects

System Maintenance Tasks

This section describes operations you need to perform on a regular basis to maintain TCP, TERM, and PROGRAM objects in a PATHMON environment. Tasks you must perform regularly include the following:

- Monitor status and performance by displaying information about your PATHMON environment and its objects
- Reconfigure, when necessary, to accommodate new application requirements, new users, and so on
- Manage exception conditions when and if they occur

For information on monitoring and maintaining your overall PATHMON environment, see the *TS/MP System Management Manual*.

Displaying Information About Pathway/iTS Objects

The PATHMON process maintains information about object configurations, object status, and operation statistics.

You can display this information using these commands:

- INFO command
- STATUS command
- STATS command

You use the STATUS command to determine the state of a PATHMON environment and its objects: running, stopped, suspended, and so on. You use the INFO command to display the current configuration of a PATHMON environment and PATHMON-controlled objects. You use the STATS command to check statistics about TCP, TERM, and SERVER resources.

For all of these commands, you can direct the display output to a text file by including the OUT option in your command. For example, this command directs output to the text file named JUNE95:

```
= INFO /OUT JUNE95/ PATHWAY
```

With the INFO command, you can use the OBEYFORM option to capture a configuration for use in future start operations. (For a discussion of the OBEYFORM option, see the *TS/MP System Management Manual*.)

This section provides several examples of the INFO, STATUS, and STATS commands, showing various syntax options for these commands and the resulting display information.

The following scenario illustrates how to use the information provided by the INFO, STATUS, and STATS commands to detect problems within your environment and determine how to reconfigure the system to rectify these problems.

A System Management Scenario

You know that users are encountering problems with a RUN PROGRAM command associated with the TCP named TCP-PROBLEM. The problem occurs more often in the morning than in the afternoon. You take these steps to investigate the problem:

1. At 9:00 a.m., you use the STATUS TCP command to check the status of TCP-PROBLEM and its TERM objects, and find that all TERM objects are in use—everyone is reading e-mail, for example.

This means that any other user attempting a RUN PROGRAM command associated with TCP-PROBLEM experiences a delay (or an error) because TCP-PROBLEM can support only a finite number of TERM objects; this number, of course, is determined by TCP-PROBLEM's configuration.

(Remember, a RUN PROGRAM command creates a temporary TERM object for that device or process.)

2. You use the INFO TCP command to find that TCP-PROBLEM supports a maximum of 25 TERM objects.
3. You use the INFO PATHWAY command to find that this PATHMON environment can support a total of 150 TERM objects; currently, 42 TERM objects are defined (25 of these are under control of TCP-PROBLEM).

The INFO PATHWAY command also tells you that this PATHMON environment can support a maximum of 20 TCPs; currently, 11 TCPs are defined, one of which is TCP-PROBLEM.

You have several options for solving this problem:

- You can ask users to stop reading e-mail at 9:00 a.m. Of course, this solution is probably not appropriate for an e-mail application, but it might be appropriate for other types of applications.
- You can increase the number of TERM objects supported by TCP-PROBLEM to fifty. Fifty users can then issue a RUN PROGRAM command to TCP-PROBLEM.
- You can associate the RUN PROGRAM command with another TCP (assuming another TCP is less busy at 9:00 a.m. and can accommodate other TERM objects). To associate the command with another TCP, copy the program description (INFO PROGRAM, OBEYFORM), edit the description to change the TCP, and add the description to the configuration.

- You can add another TCP object that might or might not be dedicated exclusively to the RUN PROGRAM command.

You must keep in mind, however, that when you make a change to your configuration, the change can affect other performance factors.

If you add 25 TERM objects to TCP-PROBLEM, for example:

- Users might encounter response time problems unless you increase the size of buffer space that TCP-PROBLEM allots for I/O operations to terminals. (That is, TCP TERMPool might be insufficient for the added terminals.)
- There might be insufficient server links available in TCP-PROBLEM.
- There might be an insufficient total number of links.
- There might be an increase in response time due to contention for the data or code areas.
- The workload might no longer be distributed evenly among CPUs in the system.

You can use the STATS command to view information about response time, memory management and allocation, link management, and so on.

For another example, if you add a TCP:

- All of the issues described for adding TERM objects could apply for adding a TCP.
- The TCP requires new swap file space on disk.
- There will be increased demand for memory in the CPU.
- There will be increased demand for other system resources, such as increased need for memory segments.

Whenever you add PATHMON-controlled objects to your configuration, you must ensure that you are still within the limits defined for your system (use the INFO command).

To ensure proper maintenance of your PATHMON environment, you must monitor its performance on a regular basis, make changes when required, and always monitor the effects of any changes that you make.

Displaying Configuration Information

The INFO command displays configuration information. Use the INFO command to display configuration information about TCP, TERM, and PROGRAM objects. For instructions on using the INFO command to display information for objects and processes running under TS/MP—for example, the PATHMON process and SERVER objects—see the *TS/MP System Management Manual*.

The INFO TCP Command

The INFO TCP command displays information about a single TCP, multiple TCPs, or all TCPs controlled by the PATHMON process.

The following example displays information about a single TCP named M6530-TCP1:

```
= INFO TCP M6530-TCP1
```

[Example 4-1](#) shows the TCP information displayed in response to the command.

Example 4-1. INFO TCP Display

```
TCP M6530-TCP1
  AUTORESTART 3
  CHECK-DIRECTORY ON
  CODEAREALEN 200000
  CPUS 12:13
  DEBUG OFF
  DUMP ON
  GUARDIAN-SWAP \SYS.$ASAP
  HIGHPIN OFF
  HOMETERM $OSP
  INSPECT ON
  MAXINPUTMSGLEN 133
  MAXINPUTMSGGS 0
  MAXPATHWAYS 20
  MAXREPLY 3000
  MAXSERVERCLASSES 50
  MAXSERVERPROCESSES 300
  MAXTERMDATA 64000
  MAXTERMS 25
  NONSTOP 1
  POWERONRECOVERY ON
  PRI 148
  PROCESS $ZCMA
  PROGRAM \*.$SYSTEM.SYSTEM.PATHTCP2
  SENDMSGTIMEOUT OFF
  SERVERPOOL 20000
  STATS OFF
  SWAP \SYS.$BABE
  TCLPROG \*.$OPER.TRANCNFG.UBIQ
  TERMBUF 3000
  TERMPPOOL 40000
```

The TCP M6530-TCP1 can have a maximum of 25 TERM objects open at the same time (indicated by the MAXTERMS parameter). M6530-TCP1 uses the SCREEN COBOL object library file named *.\$OPER.TRANCNFG.UBIQ to locate the screen programs for its terminals (indicated by the TCLPROG parameter). * is a generic name for the node on which the PATHMON process is currently running.

In case of an abnormal termination, the PATHMON process attempts to restart M6530-TCP1 three times within a fixed 10-minute interval (indicated by the AUTORESTART parameter); the primary and backup processes for this TCP run in CPUs 12 and 13, respectively (indicated by the CPUS parameter).

For a complete description of all the TCP attributes viewable through the INFO TCP command, see the description of the SET TCP command in [Section 9, Terminal Control Process \(TCP\) Commands](#).

The INFO TERM Command

The INFO TERM command displays information about a single TERM object, multiple TERM objects, or all TERM objects described in the PATHMON configuration file.

The following example displays information about two TERM objects named TAREQ1 and TAREQ2:

```
= INFO TERM (TAREQ1, TAREQ2)
```

[Example 4-2](#) shows the information displayed for the two TERM objects.

Example 4-2. INFO TERM Display

```
TERM TAREQ1
  AUTORESTART 3
  BREAK OFF
  DIAGNOSTIC OFF
  ECHO ON
  EXCLUSIVE OFF
  FILE \PARIS.$TSCH.#TAREQ
  INITIAL INIT
  INSPECT OFF
  IOPROTOCOL 0
  MAXINPUTMSGs 0
  TCLPROG \*.$OPER.TRAING.CUST
  TCP TAREQ-TCP1
  TMF ON
  TRAILINGBLANKS ON
  TYPE CONVERSATIONAL:0
TERM TAREQ2
  AUTORESTART 3
  BREAK OFF
  DIAGNOSTIC OFF
  ECHO ON
  EXCLUSIVE OFF
  FILE \PARIS.$TSCH.#TAREQ
  INITIAL INIT
  INSPECT OFF
  IOPROTOCOL 0
  MAXINPUTMSGs 0
  TCLPROG \*.$OPER.TRAING.CUST
  TCP TAREQ-TCP1
  TMF ON
  TRAILINGBLANKS ON
  TYPE CONVERSATIONAL:0
```

The two TERM objects have identical configurations.

The TERM objects are conversational-mode terminals (indicated by the TYPE parameter). INIT is the name of the SCREEN COBOL program unit that the TERM objects run on startup (indicated by the INITIAL parameter). The TCP named TAREQ-TCP1 controls the TERM operations (indicated by the TCP parameter).

For a complete description of all the TERM attributes viewable through the INFO TERM command, see the description of the SET TERM command in [Section 10, TERM Commands](#).

The INFO PROGRAM Command

The INFO PROGRAM command displays information about a single PROGRAM object, multiple PROGRAM objects, or all PROGRAM objects described in the PATHMON configuration file.

The following example displays information about the PROGRAM object named TMANAGER:

```
= INFO PROGRAM TMANAGER
```

[Example 4-3](#) shows the information displayed in response to the command.

Example 4-3. INFO PROGRAM Display

```
PROGRAM TMANAGER
  ERROR-ABORT ON
  OWNER \SYS.30,1
  SECURITY "N"
  TCP TRAN-TCP1
  TMF ON
  TYPE T16-6520 (INITIAL TMANAGER-DRIVER,
    TCLPROG \*. $OPER.TRANCNFG.CUST)
```

The TMANAGER program is associated with the TCP TRAN-TCP1 (indicated by the TCP parameter).

When starting the application, the TCP executes the SCREEN COBOL program unit TMANAGER-DRIVER (indicated by the INITIAL parameter), which is stored in the SCREEN COBOL object library file named *. \$OPER.TRANCNFG.CUST (indicated by the TCLPROG parameter). The TMANAGER program runs on a 6520 terminal (indicated by the TYPE parameter).

For a complete description of all the PROGRAM attributes viewable through the INFO PROGRAM command, see the description of the SET PROGRAM command in [Section 11, PROGRAM Commands](#).

Displaying Status Information

You can display the status of an object, error messages pertaining to the object, and other associated information using the STATUS command. Use the STATUS command to display information about TCPs and TERM objects. For instructions on using the STATUS command to display information for objects and processes running under TS/MP—for example, the PATHMON and LINKMON process and SERVER objects—see the *TS/MP System Management Manual*.

Various levels of detail can be reported using the STATUS command. For instance, the STATUS TCP command features an option for reporting the status of terminals controlled by the TCP.

The STATUS TCP Command

The STATUS TCP command displays status for a single TCP, multiple TCPs, or all TCPs defined in a PATHMON environment.

The following example displays status for all TCPs:

```
= STATUS TCP *
```

[Example 4-4](#) shows the information displayed in response to the command.

Example 4-4. STATUS TCP Display

TCP	STATE	ERROR	INFO	PROCESS	CPUS
DW-TCP1	STOPPED			\SYS.\$ZDWA	12:13
M6530-TCP1	RUNNING			\SYS.\$ZCMA	12:13
M6530-TCP2	RUNNING			\SYS.\$ZCMB	13:14
M6530-TCP3	RUNNING			\SYS.\$ZCMC	14:13
M6530-TCP4	RUNNING			\SYS.\$ZCMD	13:12
M6530-TCP5	RUNNING			\SYS.\$ZCME	12:13
M6530-TCP6	RUNNING			\SYS.\$ZCMF	13:14
TAREQ-TCP1	RUNNING			\SYS.\$ZTAA	14:13
TAREQ-TCP2	RUNNING			\SYS.\$ZTAB	13:14
TRAN-TCP1	RUNNING			\SYS.\$ZTTA	12:13
TRAN-TCP2	RUNNING			\SYS.\$ZTTB	1:2

The display shows that nearly all TCPs in this PATHMON environment are running. The display also shows the *Guardian* operating environment process name of each TCP and the current primary and backup CPUs for each TCP. The ERROR and INFO columns provide information about any system errors associated with a TCP.

You can also include state information (running, stopped, or suspended) in your command. The STATE option is very useful for problem detection and management. For example, the following command displays status only for those TCPs that are stopped:

```
= STATUS TCP *, STATE STOPPED
```

[Example 4-5](#) shows the information displayed in response to the command.

Example 4-5. STATUS TCP With STATE Option

TCP	STATE	ERROR	INFO	PROCESS	CPUS
DW-TCP1	STOPPED				

You can view the status of an external TCP by specifying the E prefix with the name of the HP NonStop system on which the external TCP is running, and the process name of the external TCP, as shown in this example:

```
= STATUS TCP E\CUPRTNO.$TCP1
```

For each TCP, you can also view the status of the terminals the TCP controls by including the **DETAIL** option in your command. For example, this command displays status information for the TCP M6530-TCP1 and all terminals under its control:

```
= STATUS TCP M6530-TCP1, DETAIL
```

[Example 4-6](#) shows the information displayed in response to the command.

Example 4-6. STATUS TCP With DETAIL Option

TCP M6530-TCP1	STATE RUNNING	ERROR	INFO	PROCESS \SYS.\$ZCMA	CPUS 12:13
TERM	WAIT		PENDING	ACCEPT	STOPMODE
077-LAM1-5BE	TERMREAD			YES	TRANSMODE
077-LAM1-737	TERMREAD			YES	
077-TP2-704	TERMREAD			YES	
077-TP3-1E0	TERMREAD			YES	
077-ZTNT-017	TERMREAD			YES	
077-ZTNT-018	TERMREAD			YES	
077-ZTNT-452	TERMREAD			YES	
077-ZTNT-475	TERMREAD			YES	
077-ZTNT-4F6	TERMREAD			YES	
077-ZTNT-670	TERMREAD			YES	
077-ZTNT-6C4	TERMREAD			YES	
077-ZTNT-6F6	TERMREAD			YES	
077-ZTNT-71D	TERMREAD			YES	
077-ZTNT-72F	TERMREAD			YES	
077-ZTNT-751	TERMREAD			YES	
116-TU2-711	TERMREAD			YES	

For each TERM object under control of TCP M6530-TCP1, the display shows the name of the TERM object and its status. In this example, all TERM objects are waiting for an **ACCEPT** statement to complete (in other words, waiting for the user to input data and press a function key). **TERMREAD** is the I/O operation required to perform an **ACCEPT** statement.

For more examples of TERM status, see the **STATUS TERM** command.

The STATUS TERM Command

The **STATUS TERM** command displays status for a single TERM object, multiple TERM objects, or all TERM objects in a PATHMON environment.

The following example displays status for two terms named TFRONT1 and TFRONT2:

```
= STATUS TERM (TFRONT1, TFRONT2)
```

[Example 4-7](#) shows the information displayed in response to the command.

Example 4-7. STATUS TERM Display

TERM	STATE	ERROR	INFO	TCP	FILE
TFRONT1	RUNNING			TAREQ-TCP1	\SYS.\$TSCH.#TFRONT
TFRONT2	RUNNING			TAREQ-TCP2	\SYS.\$TSCH.#TFRONT

The display shows the name and state of each TERM object; the name of the TCP that controls the TERM object; and the file name of the TERM object. The ERROR and INFO columns provide information about any system errors associated with a TERM object.

For each TERM object, you can also view the status of the program unit currently executing on a TERM object by including the DETAIL option in your command. For example, this command displays status information for TERM TFRONT1 and the program unit running on TFRONT1:

```
= STATUS TERM TFRONT1, DETAIL
```

[Example 4-8](#) shows the information displayed in response to the command.

Example 4-8. STATUS TERM With DETAIL Option

TERM	STATE	ERROR	INFO	TCP	FILE
TFRONT1	RUNNING			TAREQ-TCP1	\SYS.\$TSCH.#TFRONT
WAIT	PENDING	ACCEPT	STOPMODE	TRANSMODE	
SERVERIO					
PU-FILE	\SYS.\$OPER.TRAND20.TAREQ				
PU-NAME	TFRONT-A02				
PU-VRSN	1				
INST-ADDR	%000022				
INST-CODE	SEND-ON-ERR				
SERVER	TRECVC				

In this example, the TERM object is waiting for a request to the server to complete. The server is communicating with the server class named TRECVC (indicated by the SERVER parameter).

\SYS.\$OPER.TRAND20.TAREQ is the name of the currently executing SCREEN COBOL program file (indicated by the PU-FILE parameter). TFRONT-A02 is the name of the currently executing program unit (indicated by the PU-NAME parameter). The number 1 indicates the version of the program unit. The INST-ADDR parameter (%000022) is the octal offset from the base of the program unit of the currently executing instruction. SEND-ON-ERROR is the name of the currently executing instruction (indicated by the INST-CODE parameter).

You can also include state information (running, stopped, suspended) in your command. For example, the following command displays status only for those TERM objects that are not running:

```
= STATUS TERM *, STATE NOT RUNNING
```

[Example 4-9](#) shows the information displayed in response to the command.

Example 4-9. STATUS TERM With STATE Option

TERM	STATE	ERROR	INFO	TCP	FILE
MTDP-ZDYN015403	STOPPED			RSC-TCP	\SYS.\$MTDP.#RSC.ZD01

The following are usage considerations when executing the STATUS TERM command:

- If you issue the STATUS TERM command to a PATHMON process on a remote CPU and a TERM on the remote CPU is inaccessible to your local PATHCOM, the command fails with an error 1024, "TERM, ILLEGAL FILE NAME (000)." If you encounter this error, try the command again using PATHCOM on the remote CPU.
- Occasionally, you might issue a STATUS TERM command for a TERM object that has been stopped by the TCP. If your STATUS TERM command executes during the brief interval before the PATHMON process is notified that the TERM object is stopped, you may receive an error 3202, "TERMINAL IDENTIFIER NOT KNOWN TO TCP." If you encounter this error, try the command again in a few seconds.

For a complete description of the information available through the STATUS TERM command, see [Section 10, TERM Commands](#).

Displaying Statistics Information

Statistics are available for display only if you use one of these commands to direct the system to collect them:

- SET TCP STATS
- CONTROL TCP STATS

You can display statistics about TCP and TERM objects using these commands:

- STATS TCP
- STATS TERM

For details on interpreting statistical data, see [Section 5, Tuning Your System Using Statistics](#).

The SET TCP STATS Command

The SET TCP STATS command, specified during TCP configuration, tells the TCP to gather statistics until you specify a CONTROL TCP STATS OFF command. For example:

```
= SET TCP TCP-A STATS ON
```

The CONTROL TCP STATS Command

The CONTROL TCP STATS command enables you to gather statistics only when you need them. You specify this command after your PATHMON environment is running, as shown in this example:

```
= CONTROL TCP TCP-A, STATS ON
= .
  .      (5 minutes go by)
  .
= STATS TERM ...
```

```
= STATS TCP ...
```

```
= CONTROL TCP TCP-A, STATS OFF
```

Once you enter the CONTROL TCP STATS OFF parameter, all statistical counters are reset to 0.

The STATS TCP Command

The STATS TCP command displays statistics about TCP operations and data space allocation for a single TCP, multiple TCPs, or all TCPs in a PATHMON environment. (This command does not display information about external TCPs that might have links to local servers.)

The following command displays information about the TCP M6530-TCP1:

```
= STATS TCP M6530-TCP1
```

[Example 4-10](#) shows the information displayed in response to the command.

Example 4-10. STATS TCP Display

```
TCP M6530-TCP1                                07 APR 1996, 09:46:47
POOL INFO:          SIZE      REQ CNT      MAX ALLOC      AVG ALLOC      CUR ALLOC
  TERMPPOOL          10008      130799          568           66           32
  SERVERPOOL        20000      193346          3876          440           38
                MAX REQ      AVG REQ
  TERMPPOOL           260           66
  SERVERPOOL          1932          440
AREA INFO:          SIZE      REQ CNT      MAX ALLOC      AVG ALLOC      CUR ALLOC
  DATA          2248704          --          67690           --          55390
  CODE          200000      940706      154728          154724          154724
                MAX REQ      AVG REQ      % ABSENT
  DATA           --          --          ---
  CODE          18952      7774           0.0
QUEUE INFO:      REQ CNT      % WAIT      MAX WAITS      AVG WAITS
  TERMPPOOL      130799          0.0           0           0.00
  SERVERPOOL      193346          0.0           0           0.00
  MEMMAN          102004          0.3           1           0.00
  LINK              1          0.0           0           0.00
  DELINK            1          0.0           0           0.00
  CHECKPOINT      15591          0.0           0           0.00
```

If you include the DETAIL option, as shown in this example, the STATS TCP command also displays statistics, including response time information, for the terminals and server classes associated with the TCP:

```
= STATS TCP M6530-TCP1, DETAIL
```

[Example 4-11](#) on page 4-12 shows the information displayed in response to the command.

Example 4-11. STATS TCP With DETAIL Option

```

TCP M6530-TCP1                                     11 APR 1996, 14:18:02
POOL INFO:          SIZE      REQ CNT    MAX ALLOC  AVG ALLOC  CUR ALLOC
TERMPool           10008      130799      568         66         32
SERVERPool         20000      193346      3876        440        38
                  MAX REQ    AVG REQ
TERMPool           260        66
SERVERPool         1932      440
.
.
.
TERM 077-LAM1-9F5                                11 APR 1996, 14:18:02
I/O INFO:          REQ CNT    MAX TSIZE  AVG TSIZE  I/O CNT
DISPLAY           61605        0         0         0
ACCEPT            61606         82        39       61606
SEND              95560        850       176       95560
REPLY              1744        166
CHECKPOINT         12220       8245
AREA INFO:        MAX SIZE    AVG SIZE    CUR SIZE
DATA              15946        14158
CODE              18952        7758      18628
RESPONSE TIME INFO (TIME VALUES IN SECS):
ACCEPT/SEND MESSAGE
SUMMARY  # MEAS    AVG RESP    MAX RESP    MIN RESP    STAND DEV
        61605      0.55      266.63      0.00        2.10
.
.
.
SERVER AUTO-LOGON                                11 APR 1996, 14:18:03
QUEUE INFO:        REQ CNT      % WAIT    MAX WAITS  AVG WAITS  % DYNAMIC
                  0          0.0         0         0.00        0.0
I/O INFO:          REQ CNT    MAX TSIZE  AVG TSIZE  I/O CNT
SEND              14342        50        50       14342
REPLY              486        485
RESPONSE TIME INFO (TIME VALUES IN SECS):
SEND TO SERVERCLASS
SUMMARY  # MEAS    AVG RESP    MAX RESP    MIN RESP    STAND DEV
        14340      0.06      4.50      0.01        0.11
.
.
.

```

If you include the **FREQTABLE** option, as shown in this example, the **STATS TCP** command generates a frequency distribution table that contains statistics for each terminal and server class associated with the TCP:

```
= STATS TCP M6530-TCP1, FREQTABLE
```

For more information about these options and the information they provide, see [Section 5, Tuning Your System Using Statistics](#).

The STATS TERM Command

The STATS TERM command displays statistics, including response time information, about TERM operations for a single TERM object, multiple TERM objects, or all TERM objects in a PATHMON environment. These statistics are gathered by the TCP that controls the TERM or TERM objects.

For example, the following command displays statistics for the terminal named TFRONT1 every 10 seconds up to five times:

```
= STATS TERM TFRONT1, INTERVAL 10, COUNT 5
```

[Example 4-12](#) shows the information displayed in response to the command.

Example 4-12. STATS TERM Display

```
TERM TFRONT1                                07 APR 1996, 09:51:19
INTERVAL 10 SECS          COUNT 1/5
I/O INFO:                REQ CNT    MAX TSIZE    AVG TSIZE    I/O CNT
  DISPLAY                61579         0           0           0
  ACCEPT                61579         82          39        61579
  SEND                  97713        850         179        97713
  REPLY                  1744         167
  CHECKPOINT            12198        8294
AREA INFO:                MAX SIZE    AVG SIZE    CUR SIZE
  DATA                15946
  CODE                 18952        7800        18628
RESPONSE TIME INFO (TIME VALUES IN SECS) :
  ACCEPT/SEND MESSAGE
    SUMMARY  # MEAS    AVG RESP    MAX RESP    MIN RESP    STAND DEV
             61578      0.56      273.52      0.00        2.23
```

The next command displays statistics for all TERM objects associated with the TCP M6530-TCP1:

```
= STATS TERM *, TCP M6530-TCP1
```

The next command displays statistics for all TERM objects in a PATHMON environment:

```
= STATS TERM *
```

If you include the FREQTABLE option, as shown in this example, the STATS TERM command generates a frequency distribution table that contains statistics for the specified TERM object (or objects):

```
= STATS TERM TFRONT1, FREQTABLE
```

For more information about the STATS TERM command and examples showing a frequency distribution table, see [Section 5, Tuning Your System Using Statistics](#).

Reconfiguring Pathway/iTS Objects

As your business needs change, requirements for your transaction processing configuration are likely to change. Adjustments are sometimes necessary to satisfy

your transaction throughput and response time requirements and to update or expand the system to provide needed resources.

In response to your changing application requirements, you might need to specify new limits or add, alter, or delete objects running under the Pathway/iTS product.

For example, as your system grows, you might need to add TERM objects to a specific TCP to support availability of a RUN PROGRAM command, add another TCP, or add an external TCP to ease the workload of the PATHMON process.

To specify changes for objects running under TS/MP, or to perform other system reconfiguration tasks such as shutting down the PATHMON process, changing backup CPUs, or changing security attributes, see the *TS/MP System Management Manual*.

All online configuration changes (ADD, ALTER, and DELETE commands) are saved in the PATHMON configuration file. Because the ADD, ALTER, and DELETE commands operate upon the PATHMON configuration file, these commands make obsolete any previously established command file used to restart your PATHMON environment.

Specifying New Limits

You can increase or decrease limits for your overall environment using the SET PATHWAY command. For example, the following command specifies a new limit for the MAXTCPS parameters:

```
= SET PATHWAY MAXTCPS 40
```

You cannot specify new global limits while PATHMON-controlled objects are running; you must first shut down the entire configuration.

For information about shutting down and starting your configuration, see the *TS/MP System Management Manual*.

Note. When specifying limits, you should always allow space for system growth. If you specify sufficient limits initially, you can avoid the need to reconfigure and cold start your system to specify new limits.

Adding, Altering, and Deleting Objects

The following subsections describe how to add, alter, and delete objects.

Certain commands that modify an object can be performed only if the object is not running. For example, you can only delete an object that is stopped. So, before you modify an object, you should be aware of its state: stopped, suspended, running, or running but subject to a pending stop request. To determine the state of an object, use the STATUS command for that particular object, as described earlier in this section.

Adding Objects

You add TCP, TERM, and PROGRAM objects using the SET and ADD commands for that particular object. Once you have configured and added an object, you can start the object using the START command for that object.

For complete information about configuring, adding and starting objects, see [Section 2, Configuring Pathway/ITS Objects](#).

Altering Objects

You can alter TCPs and TERM, and PROGRAM objects by entering the ALTER command for the specified object type.

Before using the ALTER command for a TCP or TERM object, you must first stop the object with the STOP command. The following example changes the screen program object library and maximum number of terminals for the TCP named TCP-PRIME, and then restarts TCP-PRIME:

```
= STOP TCP TCP-PRIME
= ALTER TCP TCP-PRIME, TCLPROG $DATA.NEWAPP.POBJ, MAXTERMS 8
= START TCP TCP-PRIME
```

Deleting Objects

You can delete TCPs and TERM, and PROGRAM objects by entering the DELETE command for the object type you wish.

In response to the DELETE command, the PATHMON process removes the object definition from the PATHMON configuration file.

Before using the DELETE command for a TCP or a TERM object, however, you must first stop the object with the STOP command. For example, to delete the TERM named TERM-050 from your system, you would enter:

```
= STOP TERM TERM-050
= DELETE TERM TERM-050
```

Changing Backup CPUs and Dump Files

Although you must stop most PATHMON-controlled objects before you change their attributes, you can change three attributes for your TCPs while they are running:

- The backup CPU for the process
- The destination file for the memory dump written by the process, if the process encounters an internal or fatal error
- The LOG file

To change the backup CPU for the TCP, you enter the CONTROL TCP command for the object you wish. For instance, to change the backup CPU to CPU 7 for the TCP named TCP1, enter the following:

```
= CONTROL TCP TCP1, BACKUPCPU 7
```

To direct the memory dumps for the TCP to files named PMDUMP and TCPDUMP, respectively, enter the following:

```
= CONTROL TCP TCP-1, DUMP ON (FILE TCPDUMP)
```

The PATHMON configuration file is updated to reflect this change.

To change the file for logging output, use the PATHCOM commands LOG1 and LOG2. For example, to specify \$0 as the log file for errors reported in tokenized event message format, and to specify the disk file LOGCOPY as a log file for both error and status change information in text format, enter:

```
= LOG1 $0, EVENTFORMAT  
= LOG2 LOGCOPY, STATUS
```

Be sure to create a disk file (using the FUP CREATE command) for logging purposes before specifying it.

Exchanging Primary and Backup CPUs

You can exchange the primary and backup CPUs used for your TCP by using the SWITCH command, as shown in this example:

```
= SWITCH TCP TCP1
```

At any time after this operation, you can also reestablish the primary CPU for the TCP (as recorded in the PATHMON configuration file) by using the PRIMARY command, as shown in the next example:

```
= PRIMARY TCP TCP1
```

Neither of these operations alters information recorded in the PATHMON configuration file.

Logging Status and Error Information

TCPs report error and status information to a log file. You can request that error and status information be formatted either as text or as tokenized event messages (managed by the Event Management Service (EMS) portion of the Distributed Systems Management (DSM) software).

You can log information to a command terminal or to a disk file; it is recommended, however, that you not specify a terminal. On a terminal, log messages are lost once they scroll off the terminal screen; also, performance is generally better to a process on a disk file than to a terminal.

For instructions on specifying a destination for TCP error and status messages, see the *TS/MP System Management Manual*.

Managing Exception Conditions

You can change various aspects of a PATHMON environment as it is running. Often you will need to change system parameters because an exception or error condition has occurred for an object, such as a TERM object or TCP. Before you can respond to an exception condition, you must know that the condition exists and what caused it.

There are two ways to detect an exception condition for a TERM object or TCP:

- Periodically check the object's status with a STATUS TERM or STATUS TCP command.
- Monitor the log file for TERM and TCP information. When it fails, a TCP generates an EMS 512 event (old format), regardless of the system configuration.

You might need to use both methods to detect the conditions you are interested in because not all exception conditions show up in both sources.

Using PATHCOM Commands

The PATHCOM commands can be issued by an operator on a regular basis to check exception conditions for TERM objects and TCPs. Or, you can develop an SPI process to issue the commands, determine if a problem exists, alert the operator to the problem, and provide corrective action to take. (Several third-party companies also provide such SPI processes.)

The following examples use PATHCOM commands. For additional information on using PATHCOM commands to monitor your system, see the *TS/MP System Management Manual*. For more information about SPI processes to manage the TS/MP and Pathway/iTS products, see the *TS/MP Management Programming Manual* and the *Pathway/iTS Management Programming Manual*.

As an example of an operator using the PATHCOM commands, assume an operator checks to determine if any of the TERM objects are not running after the environment is started. The operator enters this command:

```
= STATUS TERM *, STATE NOT RUNNING
```

The STATUS command returns this information:

TERM	STATE	ERROR	INFO	TCP	FILE
TERM-1-2	SUSPENDED	3017	14	TCP-2	\SKY.\$TT2.X33
TERM-4-5	SUSPENDED	3017	14	TCP-5	\SKY.\$TT5.X45

From this information, the operator sees that two TERM objects are not running. The error codes under the ERROR and INFO columns help solve the problem. Error code 3017 is a PATHMON error that can be displayed with the HELP command:

```
= HELP 3017
    PATHTCP ERROR - *3017* ERROR DURING TERMINAL OPEN
```

Error code 14 is a file-system error that can be displayed with the TACL ERROR command:

```
11> ERROR 14
014 device does not exist
```

From these error codes, the operator determines that the file names of the devices associated with the TERM objects do not exist. To change the device file names for the TERM objects, these commands are used:

```
= ABORT TERM *, STATE SUSPENDED
TERM-1-2, STOPPED
TERM-4-5, STOPPED
= ALTER TERM-1-2, FILE $TTB.X56
= ALTER TERM-4-5, FILE $TTB.X70
= START TERM *
TERM-1-2, STARTED
TERM-4-5, STARTED
```

As a final check, STATUS TERM is used again:

```
= STATUS TERM *, STATE NOT RUNNING
=
```

Because no information is returned from the command, the operator knows that all TERM objects are running.

Using EMS Event Messages

Event messages for TCP and TERM objects are an important source of detecting exception conditions for these objects. Because of the large quantity of events often generated, it is not easy or efficient for an operator to read the event message log to detect certain exception conditions.

As an alternative to having an operator read the event messages, you can:

- Use an EMS printing distributor with filtering logic to sort out the messages the operator should take immediate action on.

For more information about printing distributors and filters, see the *EMS Manual*.

- Develop an SPI process to monitor the event messages for exception conditions to take action on.

This process can be designed to determine the cause of the problem, based on the event message, and then report the condition to the operator, along with corrective action to take.

For more information about SPI processes and event messages for the TS/MP and Pathway/iTS products, see the *TS/MP Management Programming Manual* and the *Pathway/iTS Management Programming Manual*.

Managing Links

A link is a connection between a link manager, such as a TCP, and a specific server process. The link is used to send a request to, and receive a reply from, a server.

Note. The Pathway environment provides two types of link managers: TCPs and LINKMON processes. TCPs manage links to server processes from SCREEN COBOL requester threads. LINKMON processes manage links to server processes from Pathsend processes and from clients and requesters running under related products such as the HP NonStop Remote Server Call/MP (RSC/MP) software and the HP NonStop TUXEDO system. For more information about LINKMON processes, see the *TS/MP System Management Manual*.

A TCP shares links among its multiple SCREEN COBOL requesters. Only one SCREEN COBOL program at a time can use a link (by performing a SEND). TCPs initiate link requests to Pathway servers.

A TCP attempts to use one of its existing links to satisfy a send request. If no such link is available, then the TCP process does one of the following:

- If the TCP can get a static link, it asks for a link immediately.
- If the TCP cannot get a static link, and no static links become available in the period of time specified in the CREATEDELAY parameter, then the TCP asks for a dynamic link.

A link is managed and owned by the PATHMON process that controls the server process. To perform link management, the PATHMON process maintains status information for TCP, TERM, and SERVER objects.

By understanding how the PATHMON process manages links and what causes dissolution of links, you can take steps to improve the performance of your system. Steps you can take to improve link performance are described in the *TS/MP System Management Manual*. Information on deletion of links appears in this subsection.

Understanding the Causes of Link Dissolution

A PATHMON process, LINKMON process, server process (indirectly), or TCP can delete a link. To learn about deletion of links by PATHMON, LINKMON, or server processes, see the *TS/MP System Management Manual*.

A TCP will delete a link to a server process when any of this situations occur:

Cause

A “DELETEDELAY” timer expires for a dynamic link.

An OPEN error occurs on a link.

An I/O error occurs on a link. (I/O errors include timeouts caused by the SET SERVER TIMEOUT value. Timeouts caused by the SERVERCLASS_SEND_ timeout value are not considered to be link errors.)

The PATHMON process sends a server class stop request.

The PATHMON process sends a forced delink request.

The PATHMON process sends an external shutdown notification

The PATHMON process returns an unexpected I/O error or close message, or the TCP detects a message protocol error.

External TCP gets an error trying to communicate with the PATHMON process

Action

Returns the dynamic link.

Returns the link.

Returns all links for this server process, allowing current I/O to complete.

Returns all links for this server process, allowing current I/O to complete.

Returns all links to the server process, allowing current I/O to complete.

Deletes all links owned by that PATHMON process, allowing current I/O to complete. Consequently, no links are returned to the PATHMON process.

Deletes all links owned by that PATHMON process, allowing current I/O to complete. Consequently, no links are returned to the PATHMON process.

Deletes all links owned by that PATHMON process, allowing current I/O to complete. Consequently, no links are returned to the PATHMON process.

Improving Performance

You can improve the performance of the PATHMON process in some instances by reconfiguring your PATHMON environment. The following subsections describe options that you might want to consider for objects running under the Pathway/iTS product. For information on other options for improving the performance of the PATHMON process, see the *TS/MP System Management Manual*.

Use External TCPs to Manage Terminals

You can distribute the workload by configuring external TCPs to handle terminals. By distributing the workload and taking advantage of parallelism, you can also decrease both startup and recovery time.

You can manage any increase in operational complexity by using automated operators.

Prevent TCLPROG File Checking

System performance is improved if TCLPROG file checking is not performed. At the TCP level, set the CHECK-DIRECTORY attribute to OFF to prevent the system from checking the TCLPROG file. To prevent the system from checking the TCLPROG file at the TERM and PROGRAM level, you must remove the TCLPROG file.

Because partitioned code is easier to manage, you must weigh the cost of reduced manageability when considering whether to improve system performance using this method.

Improving Ready-Time-to-Busy-Time Ratio

The ready-time-to-busy-time ratio can have a significant effect on a transaction's elapsed time.

For example, if the TCP CPU time for a transaction is 300 ms and the ready-time-to-busy-time ratio is 6 to 1, the elapsed time in the TCP for this transaction is 1.8 seconds. This applies to all processes in the transaction's execution path.

A high ratio is usually an indication of an ever-busy CPU for that process, or of inappropriate priority settings. Remember that raising the process' priority is likely to increase the ready-time-to-busy-time ratio for some other process or processes in that CPU.

You can use these products to determine the ready-time-to-busy-time ratio:

- Measure product
- HP Tandem Performance Data Collector (TPDC)
- FlowMap product

For information about setting process priority, see [Section 2, Configuring Pathway/iTS Objects](#).

Information to Include When Reporting Problems

When requesting your service provider's help in resolving a problem with the Pathway/iTS product, you can greatly enhance the service provider's ability to help you by providing information about your Pathway environment. Information to be collected for Pathway/iTS components is covered in this subsection. For a description of information you need to collect for TS/MP objects and processes, see the *TS/MP System Management Manual*.

TCP-Specific Problems

Collect this information if you detect a problem with a TCP.

When describing the problem to your service provider, include details of how to reproduce the problem by navigating to particular screens and pressing particular keys.

TCP Dump

When a TCP detects an inconsistency in its control tables, the TCP dumps automatically if the dump feature is enabled. To enable the TCP dump feature, perform the following:

```
>PATHCOM $<pm>
=CONTROL <tcpname>, DUMP ON (FILE <tcpdump>)
=EXIT
```

where *pm* is the name of your PATHMON process, *tcpname* is the name of the TCP and *tcpdump* is name of the file where you want the TCP to dump errors.

If the TCP tables (showing, for example, terminal states or TCP states) appear to be in error, you can force a dump by performing the following:

```
>PATHCOM $<pm>
=CONTROL <tcpname>, DUMPMEMORY <option> (FILE <pmdump>)
=EXIT
```

where *option* is PRIMARY, BACKUP or BOTH. Use PRIMARY unless a TCP is reporting errors from its backup.

If the TCP primary process is running in the CPU defined for the TCP backup process because of a SWITCH command or some other processor change, the BACKUPCPU attribute causes the PATHMON process to change the PATHMON configuration file so that it contains the current CPU numbers for both the primary and backup TCPs.

If a TCP was configured with the NONSTOP attribute set to 0, changing the BACKUPCPU attribute only changes the backup CPU value in the PATHMON configuration file; no processor change occurs.

When you cool start a PATHMON environment after using the CONTROL TCP command, the TCPs start according to the changes recorded in the PATHMON configuration file.

If you request the help of your service provider in analyzing a problem, the representative will likely require a dump file. It is therefore recommended that you always set the DUMP option to ON for production systems.

The DUMPMEMORY option is not a substitute for setting the DUMP option to ON. DUMPMEMORY is primarily useful in a controlled troubleshooting situation where you need to take a snapshot of the TCP's internal state at a particular time - before it encounters a fatal error.

PATHMON Configuration and TCP Object Information

Make the PATHCTL file available to your service provider. Also, use PATHCOM commands to display information about the TCP at the time of the problem, as follows:

```
>PATHCOM $<pm>  
=INFO tcp  
=EXIT
```

where *pm* is the name of your PATHMON process and *tcp* is the name of a TCP.

Requester Object Files and SCREEN COBOL or User Conversion Source Code

Include the requester object (POBJ) file or files when collecting data about a TCP problem. Also provide the SCREEN COBOL source code, including copy libraries, if the problem deals with TCP logic handling within a program unit.

If the TCP problem involves user conversions, include the source code for the user conversion routine.

Terminal-Specific Problems

Collect the following information if you detect a problem with a terminal or other TERM object:

- The PATHMON configuration file, PATHCTL
- A PATHMON process dump and a TCP dump
- Status detail; at the system prompt, perform the following:

```
STATUS TERM *, DETAIL
```
- The requester source code and object module or modules
- Terminal or device type and access method

SCREEN-COBOL-Specific Problems

Collect the following information if you detect a problem with SCREEN COBOL:

- Server source code and object modules for run-time errors.
- For problems involving SEND processing, descriptions of screens to access and function keys to press to execute the SEND statement.
- The SCOBOLX, COBOLX2, and SYMSERV object files used to compile the requester.
- The SCREEN COBOL source code files for the requester.
- The SCREEN COBOL object code (POBJ) file or files.

- All copy libraries referenced in the source file or files.
- If applicable, a compiler listing showing the problem.
- Error messages written to the terminal but not included in the compiler listing.

SCUP-Specific Problems

If you detect a problem with SCUP, collect the relevant object code (POBJ) files for review.

Keeping Development and Production Separate

It is recommended that you do not mix your development and production environments. Ideally, you should have separate PATHMON environments for your development and production environments.

If you do not maintain separate environments, you might jeopardize the availability of your production environment. For example, suppose there are problems with a Guardian server that indicate the developer must use the HP *Inspect* symbolic debugger to debug the server. During this time, the server is unavailable, and any processes attempting to establish links to the sever could encounter an unacceptable delay. Similarly, a TCP that handles both production and development SCREEN COBOL requesters might not be able to respond to a user if a developer is making changes based on development requirements.

If you cannot maintain totally separate development and production environments, try to keep as much of your production environment separate as you can. It is highly recommended that you maintain separate sets of development and production server classes. It is also desirable to define separate TCPs for development and production.

Certain types of errors, such as timeout errors, can also be returned to requesters when the associated servers in a Pathway application are being debugged. For further information about these errors, see the section on servers in the *TS/MP Pathsend and Server Programming Manual*.

Sending Messages to Users

You can send messages to application users using the TELL TERM command. These messages can contain any kind of information, ranging from informative text to urgent requests.

(TELL TERM is a PATHCOM command. For more information on using PATHCOM commands to manage your PATHMON environment, see the *TS/MP System Management Manual*.)

You can direct these messages to selected users (addressing their terminals by TERM object name), or to all users accessing your application. Devices represented by both configured TERM objects and temporary TERM objects can receive these messages.

For example, the following command sends a message about an orientation meeting to a new user sitting at the device configured through TERM-06:

```
= TELL TERM TERM-06, "New Employee Orientation at 1:00 p.m."
```

The next command sends a message about a planned computer system upgrade to all users of your application:

```
= TELL TERM *, "Two New CPUs to be Added Tonight!"
```

In response to the TELL command, the PATHMON process assigns your message a unique identifying number and queues the message for delivery.

Your message does not interrupt the receiving user's normal operation at the input-output device or process; it is not displayed until the user completes interaction with the current screen. Furthermore, this message can only be displayed if the message display capability is enabled in the screen program run for the device or process.

The PATHMON process automatically deletes a message when all destination TERM objects acknowledge its receipt.

You can display the message number and text of all pending messages by entering:

```
= INFO TELL *
```

The TELL message number assigned by the PATHMON process enables you to see the message in specific commands. You will need to use this number, for instance, to delete TELL message text through PATHCOM. You can delete any individual pending message by using the DELETE command followed by the message number, for example:

```
= DELETE TELL 26
```

The DELETE command cancels delivery of a pending message. You cannot, however, issue a global DELETE command that removes all pending messages.

Migrating Pathway/iTS Objects to a Different System

Careful preparation is required to switch objects in a PATHMON environment to a different system, either in the event of failure or as part of a planned migration. The *TS/MP System Management Manual* provides general considerations for migrating an entire PATHMON environment to a different system. The node independence feature of the PATHMON process, described in that manual, provides assistance for such migrations.

The following paragraphs provide information specific to migrating Pathway/iTS objects to a different system.

Even with a node-independent PATHMON environment, a number of migration considerations arise. The node-independent designation can provide a smooth migration for disk files and process names. However, you might have to change parts of device names manually even if the node portion of the name is independent. For example, if you are migrating terminals or printers, you will either need to manually change the device names or ensure that terminal and printer devices with the same names are attached to the new node. Other attributes need careful review as well—for example, CPU and user ID specifications. [Table 4-1](#) lists Pathway/iTS object attributes that might need to be changed with the ALTER command when migrating your application.

Table 4-1. Migration Considerations: Pathway/iTS Object Attribute Values

Object or Process	Attribute	Considerations/ Recommendations
TCP	CPUS <i>pricpu:backupcpu</i>	Make sure that CPUs on new system have same numbers or change manually
	HOMETERM <i>termname</i>	Make sure that a terminal with same name exists on new system or change manually
TERM	FILE <i>filename</i>	Make sure that a device with same name exists on new system or change manually
PROGRAM	OWNER <i>owner-id</i>	Make sure that user ID is known to new system or change manually
	PRINTER <i>filename</i>	Make sure that a device with same name exists on new system or change manually

Note. The SWAP and GUARDIAN-SWAP attributes are not included in [Table 4-1](#) because TCP swap space is now handled by the Kernel-Managed Swap Facility (KMSF) and the SWAP and GUARDIAN-SWAP values are no longer used. However, if you do specify device names for these attributes, you must ensure that the named devices exist; otherwise, an error occurs. In a node-independent environment, it is recommended that you do not specify values for these attributes.

You should also consider whether your SCREEN COBOL requesters—perhaps those run by TCPs on remote systems—use hard-coded node names in requests to server processes that belong to the application you want to migrate. If screen application requests have not been coded for node independence, the applicable code must be changed manually.

Tuning Your System Using Statistics

Statistics and System Tuning

This section describes the HP NonStop Pathway/iTS terminal control process (TCP) and the statistics that the TCP produces. You can use the statistics generated by transaction processing functions in the PATHMON environment to detect bottlenecks in TCP processing. Based on careful analysis of these statistics, you can reconfigure your PATHMON environment to eliminate some performance degradations.

For information on gathering other statistics about your PATHMON environment and for instructions on reconfiguring the environment, see the *TS/MP System Management Manual*.

This section is divided into two parts:

- The first part describes the TCP and the tasks it performs.
- The second part describes the statistics that reflect how well the tasks are being performed.

Some factors affecting system performance are not represented in these statistics. Such factors generally involve contention for system resources. Examples are:

- Paging by the operating system incurred by the TCP
- The ability of the TCP to handle I/O terminations promptly

TCP Tasks

The TCP allocates internal resources and maintains queues for many of its operations and tasks, such as:

- Link management
- Memory management
- Checkpointing
- Gathering statistics

The following subsections describe the TCP tasks and the queues associated with these tasks.

Link Management

Links enable communication between requester programs and server programs. The TCP coordinates the sharing and dissolving of links between SCREEN COBOL requester programs and server programs.

The following list describes the properties of links:

- A link results in a single open of a server with a nowait depth of 1. This open is available for use by multiple requesters on a serial basis.
- A link can be used for one send operation at a time.
- A link between the TCP and a server process is busy when it is being used by a terminal task to communicate with a server; otherwise, it is available.
- Links become available when:
 - A busy link is freed by its user.
 - A new link to the server class is requested from and granted by the PATHMON process.
- For link management, two queues are maintained: LINK and DELINK.

Establishing Links

A terminal task requests an available link by executing a SCREEN COBOL SEND statement. (The link becomes available at the end of the SEND statement.)

When the TCP interprets the SEND command, it checks to see if an existing link is available:

- If an existing link is available, the TCP allocates the link to the requesting task and SEND processing continues.
- If an existing link is not available, the terminal task of the TCP can do one of these:
 - Join the link queue associated with that server class.
 - While on this queue, the terminal task is blocked from execution. When an available link is assigned to the task, the task is activated.
 - Request a new link from the PATHMON process.

Requesting a New Link

When a terminal task requests a new link, the task joins both the link queue and the server-class wait queue inside the TCP:

1. The TCP specifies a timeout value for the LINK queue.
2. After the timeout expires, the TCP issues the link request to the PATHMON process.
3. The task remains on the LINK queue until:
 - The PATHMON process establishes the link.
 - The PATHMON process denies the request for a new link.

When a link to the server class is available, the task is activated and removed from the queues. If a link to the server class is denied, control is passed to a SCREEN COBOL error handling routine (if present); otherwise, the TERM is suspended.

To specify a timeout value, the TCP uses the value specified with the SET SERVER CREATEDELAY command, but only if the number of existing links to the server class is greater than 0 and greater than the maximum number of static links available to the TCP for that server class. Otherwise, the TCP sets the time-out value to 0.

The CREATEDELAY parameter specifies the maximum amount of time a link manager waits to use an established link to a server class before requesting a new link from the PATHMON process.

Granting a New Link

For the PATHMON process to grant a new link, the TCP must have enough resources to support the new link:

- The number of links from the TCP to all server processes at the time of the request is fewer than that specified by the SET TCP MAXSERVERPROCESSES command.
- The number of server classes represented by processes that are linked to the TCP is fewer than that specified by the SET TCP MAXSERVERCLASSES command.

If the PATHMON process grants a new link, it designates the link as either static or dynamic:

- A static link is usually not dissolved unless an error occurs over that link.

Initially, you set the maximum number of static links for a server class with the SET SERVER NUMSTATIC command. Later, after examining information returned from the PATHMON process, you can adjust the maximum number of static links.

- A dynamic link is dissolved after it remains idle for a specified time period.

If no terminal tasks are waiting, the TCP sets a timeout value for the dynamic link—using the value specified with the SET SERVER DELETEDELAY command—and places the link on the TCP DELINK queue.

The DELETEDELAY parameter specifies the maximum amount of time a link between a link manager and a dynamic server can remain idle before the link manager automatically closes the server and informs the PATHMON process.

If the timeout expires, the TCP sends the delink request to the PATHMON process, which dissolves the link.

Dissolving Links

The TCP dissolves a link as follows:

1. The TCP specifies a timeout value for the link and places the link on the DELINK queue.
2. After the timeout expires, the TCP removes the link from the DELINK queue and relays the request to the PATHMON process.

Links can be removed from the DELINK queue to satisfy a request for a link from a terminal task.

Memory Management and Allocation

When a terminal task is started (via the START TERM or RUN PROGRAM command), the TCP fetches SCREEN COBOL object code from the disk to its extended data segment. The TCP uses its extended data segment for the SCREEN COBOL code segments and for terminal context data. The TCP ensures that the proper code segments are resident in memory, as needed. Additionally, the TCP allocates buffer space for I/O operations.

- The TCP allocates memory for storage, data, and code area.

Storage Area

During interpretation of a SCREEN COBOL program, the TCP is responsible for dynamically allocating and deallocating memory space for terminal and server I/O buffers.

Each TCP maintains two storage pools in memory: TERMPOOL and SERVERPOOL. These pools provide buffer space as follows:

- TERMPOOL
Provides buffer space for I/O operations to terminals.
- SERVERPOOL
Provides buffer space for I/O operations to servers.

A wait queue is associated with each pool. If enough pool space is available, the request is granted and the requesting task proceeds. Otherwise, the task joins the wait queue until enough pool space is available.

Data Area

The TCP allocates terminal data area in an extended segment of memory.

For each terminal task, the TCP allocates two slots of addresses in its data area:

- SLOT0 is used to simulate one execution stack for the executing SCREEN COBOL program units. This stack grows and shrinks during program execution.
- SLOT1 is used to hold the last checkpoint image.

Slot sizes are determined primarily by the value specified with the SET TCP MAXTERMDATA command. The sizes are approximate because the TCP rounds up to a page boundary (2048 bytes).

Code Area

The TCP allocates its terminal code area in the same extended segment as its data area.

A compiled SCREEN COBOL program unit is composed of executable code and a sequence of screen descriptions. The executable code and screen descriptions are defined as code segments as follows:

- Executable code segment (segment 0)
This segment contains the executable code and is always defined for an active task.
- Current base screen segment (segment 1)
This segment contains the current base screen. This is the screen used in the most recent DISPLAY BASE command. Segment 1 must be defined if segment 2 is defined: the user must first display the base screen that the overlay is mapped onto.
- Current overlay screen (segment 2)
This segment contains the current overlay screen. This is the screen used in the most recent operation affecting an overlay area of the current base screen.

When a terminal task is about to execute:

- The executable code (segment 0) is present in the TCP's extended segment.
- The screen descriptions are fetched when they are referenced—not when the program unit containing them is invoked.

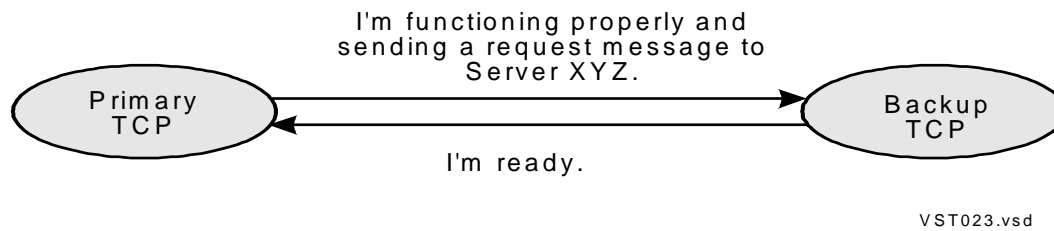
When the task is not executing, its code segments can be overlaid by other code segments.

Checkpointing

Checkpointing refers to the communication between the primary and backup processes of a TCP process pair.

During processing, the primary TCP keeps the backup TCP informed of what it is doing—for example, sending a request, receiving a reply, and so on—with checkpoint messages, as shown in [Figure 5-1](#). Consequently, if the primary process fails, the backup process has enough information to take over and continue.

Figure 5-1. TCP Checkpointing



Checkpointing occurs as follows:

1. A TCP task requests service by placing itself on the request queue associated with checkpointing. While waiting for service, the TCP task cannot execute.
2. When the service completes, the TCP task is removed from the request queue.

If TMF is being used, the TCP checkpoints the task's data area at the BEGIN TRANSACTION and END TRANSACTION statements. If the TMF facility is not installed, the TCP checkpoints before and after a SEND request to a server.

Gathering Statistics

A TCP gathers statistics about the tasks it performs, the terminals under its control, and the servers it has open. These statistics are reported to the PATHMON process.

Thus, a TCP gathers TCP, TERM, and SERVER object statistics. Details about statistics for TCP and TERM objects are included in separate subsections later in this section. For information about SERVER object statistics, see the *TS/MP System Management Manual*.

TCP Statistics

TCP statistics provide information about the following:

- Storage pools (POOL INFO)
- Memory segments (AREA INFO)
- Queues associated with the storage pools, memory management, link management, and checkpointing (QUEUE INFO)

[Example 5-1](#) shows a sample set of TCP statistics. These statistics are described in the subsections that follow.

Example 5-1. Sample TCP Statistics

```

TCP TCPX2                                02 MAY 1996, 14:03:02
INTERVAL 5 SECS      COUNT 2/5
POOL INFO:           SIZE  REQ  CNT  MAX  ALLOC  AVG  ALLOC  CUR  ALLOC
  TERMPPOOL           40962    132    1596      280      12
  SERVERPOOL          55278      0      0      0      0      0
                MAX REQ  AVG REQ
  TERMPPOOL           1596      280
  SERVERPOOL           0        0
AREA INFO:           SIZE  REQ  CNT  MAX  ALLOC  AVG  ALLOC  CUR  ALLOC
  DATA              225280      366    5336      3406
  CODE               65536      366    10200     5038    10196
                MAX REQ  AVG REQ  % ABSENT
  DATA
  CODE              8328    3802      1.3
QUEUE INFO:    REQ  CNT  % WAIT  MAX  WAITS  AVG  WAITS
  TERMPPOOL      816      0.0      0      0.00
  SERVERPOOL     3095      0.0      0      0.00
  MEMMAN         1176      1.3      1      0.01
  LINK            1      0.0      0      0.00
  DELINK          1      0.0      0      0.00
  CHECKPOINT     320      0.0      0      0.00

```

The TCP does not generate values for these fields: DATA REQ CNT, DATA AVG ALLOC, DATA MAX REQ, DATA AVG REQ, and DATA % ABSENT.

POOL INFO

The POOL INFO section in the statistics display provides information about the two storage pools in memory: TERMPOOL and SERVERPOOL.

[Example 5-2](#) shows only the POOL INFO section of the sample TCP statistics.

Example 5-2. Sample TCP Statistics for POOL INFO

```
TCP TCPX2                                02 MAY 1996, 14:03:02
INTERVAL 5 SECS      COUNT 2/5
POOL INFO:           SIZE REQ CNT  MAX ALLOC  AVG ALLOC  CUR ALLOC
  TERMPOOL           40962      132    1596      280      12
  SERVERPOOL         55278       0      0        0        0
                MAX REQ  AVG REQ
  TERMPOOL           1596      280
  SERVERPOOL         0        0
                .
                .
                .
```

The [Table 5-1](#) lists the statistics for the storage pools and what they indicate.

Table 5-1. TCP Statistics for POOL INFO

Statistic	For TERMPOOL and SERVERPOOL, Indicates:
SIZE	Maximum number of bytes available in the pool, as defined by the SET TCP TERMPOOL and SET TCP SERVERPOOL commands. (The current number of free bytes available in each pool area changes as the TCP allocates and reclaims I/O buffers for terminal and server operations.)
REQ CNT	Number of requests the TCP receives for buffer allocation.
MAX ALLOC	Greatest number of bytes allocated from the pool area at one time for all I/O buffers. (Because the TCP requires buffer space for its own operations, the buffer allocations under TERMPOOL may be larger than the size of I/O buffers alone. The size used for internal operations may vary by terminal type.)
AVG ALLOC	Average number of bytes allocated from the pool area for all I/O buffers.
CUR ALLOC	Current number of bytes allocated from the pool area for all I/O buffers.
MAX REQ	Size of the largest request for buffer allocation. (The number actually displayed is four bytes larger than the maximum buffer requested because the system adds four bytes to be used as boundary tags.)
AVG REQ	Size of the average request for buffer allocation.

SIZE

SIZE indicates the maximum number of bytes available in each pool, as determined by the TCP's configuration. If this number is consistently low (the value for MAX ALLOC is near the value for SIZE), terminal tasks are using most of the available pool space; in this case, you might want to increase the size of the pools.

You can derive the currently available pool space by using this formula:

$$\text{FREEPOOL} = \text{SIZE} - \text{CUR ALLOC}$$

You can use this formula to determine whether to increase pool size:

```
IF SIZE - MAX ALLOC < AVG REQ or MAX REQ
  THEN increase pool size
```

These allocations use memory to improve performance. If there are memory shortages, you might consider reducing these allocations.

To determine a sufficient size for each pool, use the guidelines described next.

TERMPOOL

TERMPOOL, specified with the SET TCP TERMPOOL command, indicates the minimum number of bytes that the TCP allocates in its data area for all terminal I/O buffers.

To determine a sufficient initial size for TERMPOOL, use this formula. (For these calculations, PATHMON-controlled PROGRAM objects count as terminals.) Note that these allocations use memory to improve performance. If there are memory shortages, you might consider reducing these allocations.

$$\text{TERMPOOL} = \text{MAXTERMS} * \text{TERMBUF}$$

TERMBUF governs the size of the output buffer. For input, the TCP locks down an appropriate size buffer in TERMPOOL without regard to the TERMBUF setting. Generally speaking, the output buffer is larger than the input buffer. If an application on your system could potentially have a larger buffer to be read in than to be written out, use the read buffer size when determining TERMBUF size.

- **MAXTERMS**
specified with the SET TCP MAXTERMS command, indicates the maximum number of terminals that the TCP can have open at the same time. To determine the current value for MAXTERMS, use the INFO TCP command.
- **TERMBUF**
specified with the SET TCP TERMBUF command, indicates the maximum number of bytes that the TCP allocates from the TERMPOOL area for terminal output buffers. If you do not specify a value for TERMBUF, the default is 1500. To determine the current value for TERMBUF, use the INFO TCP command. TERMBUF size is not a factor in READs or SEND messages. The following are guidelines for specifying the size of TERMBUF:

- In general, TERMBUF should be large enough to hold the largest display sent to a terminal. Although this might oversize TERMPOOL, the excess space is held only for a short time, so system performance is not adversely affected. The excess space is not used because, generally, the terminal is in a read state rather than a write state.
- Defining too small a value for TERMBUF might increase the number of output operations necessary to perform certain functions (for example, execution of the SCREEN COBOL DISPLAY BASE statement).
- Defining too large a value for TERMBUF might decrease the number of terminals able to operate simultaneously or increase the number of times an I/O operation is delayed until space for a buffer becomes available.

To adjust the size of TERMPOOL, use the STATS TCP display information (MAX ALLOC) to determine the appropriate size. You can use this formula:

$$\text{TERMPOOL} = \text{MAXTERMS} * (\text{MAX REQ} + 2)$$

SERVERPOOL

SERVERPOOL, specified with the SET TCP SERVERPOOL command, indicates the number of bytes that the TCP allocates for requests and replies between SCREEN COBOL programs and server processes.

If SERVERPOOL space is insufficient, transactions must wait for the completion of previous transactions before proceeding.

To determine sufficient size for SERVERPOOL, you need to estimate the number of buffers required for the active transactions. (Every active transaction requires one buffer.)

You can use these formulas to estimate a sufficient size for SERVERPOOL. This formula gives you an upper limit for the size of SERVERPOOL:

$$\text{SERVERPOOL} = \text{MAXREPLY} * \text{MAXSERVERS}$$

This formula gives you a less memory-intensive estimate:

$$\text{SERVERPOOL} = \text{MAXREPLY} * \text{MAX-TPS} * \text{EST-RESP-TIME} * \text{SAFETY-FACTOR}$$

- **MAXREPLY**
specified with the SET TCP MAXREPLY command, indicates the maximum number of bytes permitted for a SEND message or a REPLY message. This value should be supplied by your application development group. To determine the current value for MAXREPLY, use the INFO TCP command.
- **MAXSERVERS**
defined with the SET SERVER MAXSERVERS command, is the maximum number of server processes that can run in a server class at the same time. To determine the current value of MAXSERVERS, use the INFO SERVER command. For information about the SET SERVER MAXSERVERS and INFO SERVER commands, see the *TS/MP System Management Manual*.

- **MAX-TPS**

indicates the maximum number of transactions per second (the peak transaction rate). You can determine MAX-TPS in several ways. For example:

- You can code your application to record this information.
- You can estimate this information by counting the number of SEND statements to a server class (where the server class has a fixed relationship to a transaction).

To determine the number of SEND statements to a server class, analyze the server statistics for I/O INFO. REQ CNT indicates the total number of SEND statements to servers in a server class.

If you do not know the maximum transaction rate, use an estimate of 1.5 to 5 times the average transaction rate.

To determine the average transaction rate, you can use the Measure product or specify the STATS SERVER, INTERVAL SECS command and use the display information (I/O COUNT). You can use this formula:

$$\text{AVGTXRATE} = \text{IO-COUNT} / \text{SECONDS}$$

- **EST-RESP-TIME**

indicates the estimated time, in seconds, that a transaction takes to complete. To determine the estimated response time for a transaction, analyze the response time statistics for both the terminal and server involved in the transaction. By adding these response times, you can estimate the response time for a transaction.

- **SAFETY-FACTOR**

provides a safety factor to your formula. As a precaution, SAFETY-FACTOR increases the size of SERVERPOOL a little more than you think is needed. As a general guideline, you can specify a value from 1.1 to 1.4 for the safety factor.

At any time, the maximum value that you should specify for SERVERPOOL is:

$$\text{SERVERPOOL} = \text{MAXTERMS} * \text{MAXREPLY}$$

To determine the current values for MAXTERMS and MAXREPLY, use the INFO TCP command.

To adjust the size of SERVERPOOL, use the STATS TCP display information (MAX ALLOC) to determine the appropriate size.

REQ CNT

For both SERVERPOOL and TERMPOOL, REQ CNT indicates the physical number of requests for buffer allocation. Some DISPLAY and ACCEPT operations require multiple physical I/O operations; consequently, there is a one-to-many relationship between these statements and pool allocation. SEND statements always require only one I/O operation; consequently, there is a one-to-one relationship between SEND statements and pool allocation.

AREA INFO

The AREA INFO section in the statistics display provides information about the data and code segments in memory. This information is primarily useful to HP internal development.

- DATA is the area of the extended data segment that the TCP allocates for context data for all of its terminals.
- CODE is the area of the extended data segment that the TCP allocates for SCREEN COBOL object code.

[Example 5-3](#) shows only the AREA INFO section of the sample TCP statistics.

Example 5-3. Sample TCP Statistics for AREA INFO

```
TCP TCPX2                                02 MAY 1996, 14:03:02
  INTERVAL 5 SECS    COUNT 2/5
.
.
.
AREA INFO:          SIZE REQ CNT  MAX ALLOC  AVG ALLOC  CUR ALLOC
  DATA             225280          5336          3406
  CODE              65536          366      10200      5038      10196
                MAX REQ  AVG REQ    % ABSENT
  DATA
  CODE              8328          3802          1.3
.
.
.
```

The [Table 5-2](#) on page 5-13 lists the statistics for the code and data areas and what they indicate.

Table 5-2. TCP Statistics for AREA INFO

Statistic	For CODE, Indicates	For DATA, Indicates
SIZE	Total number of bytes allocated to the code area for all SCREEN COBOL object code	Total size of all slots for all terminal tasks
REQ CNT	Number of requests for defined code segments	N.A.
MAX ALLOC	Largest number of bytes allocated from the code area at any given time	Largest number of bytes used in the data area at one time by all terminals
AVG ALLOC	Average number of bytes allocated from the code area at one time	N.A.
CUR ALLOC	Current number of bytes allocated from the code area	Current number of bytes used in the data area by all of the active terminals
MAX REQ	Largest number of bytes requested for SCREEN COBOL code space	N.A.
AVG REQ	Average number of bytes requested for SCREEN COBOL code space	N.A.
% ABSENT	Percentage of the code segments that are not in the code area at the time the TCP is ready to execute a terminal task	N.A.

SIZE

For the code area, SIZE indicates the total number of bytes allocated for all SCREEN COBOL code. You define this value with the SET TCP CODEAREALEN command.

The value you specify depends on the number and the size of the screen programs in your applications. The code area should be large enough to hold all of the SCREEN COBOL code. (Sizes of 2 MB and larger are not unusual.)

Two possible methods for estimating code area are:

- You can estimate a value by using the size of the SCREEN COBOL programs (the screen data plus the RUNUNIT SIZE listed for these programs compiled with MAP) that make up the applications for this TCP. Note that RUNUNIT SIZE does not include base and overlay screens.

Not all of the screen programs are executed all of the time; the most regularly used programs determine the amount of code space most often required.

- You can specify a value larger than required (choose a large size, such as three or four megabytes).

This costs extra disk space for the swap file, but prevents the TCP from performing unnecessary code fetching.

(The SCREEN COBOL code area is not shared between multiple TCPs within a CPU. Consequently, large code areas might create the potential for increasing the memory pressure within the operating system. However, it is more efficient for the system to perform code fetching than the TCP.)

To adjust the size of the code area, run the TCP with STATS ON and use the STATS TCP display information to determine the appropriate size.

REQ CNT

For the code area, REQ CNT indicates the number of requests for defined code segments. A terminal task can request at most three defined code segments: executable pseudocode, base screen definitions, and overlay definitions. Each defined segment fetched is counted separately.

(The number of defined segments not present in the code area is used in the calculation of the % ABSENT entry.)

This information is primarily useful to HP internal development.

MAX REQ

For the code area, MAX REQ indicates the largest number of bytes requested for SCREEN COBOL code space. This request includes the total number of bytes for, at most, three code segments: executable pseudocode, base screen definitions, and overlay definitions.

QUEUE INFO

The QUEUE INFO section in the statistics display provides information about the queues associated with this storage pools and tasks:

- TERMPOOL
- SERVERPOOL
- Memory management (MEMMAN)
- Link management (LINK and DELINK)
- Checkpointing (CHECKPOINT)

[Example 5-4](#) on page 5-15 shows only the QUEUE INFO section of the sample TCP statistics.

Example 5-4. Sample TCP Statistics for QUEUE INFO

```
TCP TCPX2                                02 MAY 1996, 14:03:02  
INTERVAL 5 SECS      COUNT 2/5
```

.

.

	REQ	CNT	% WAIT	MAX WAITS	AVG WAITS
QUEUE INFO:					
TERMPPOOL		816	0.0	0	0.00
SERVERPOOL		3095	0.0	0	0.00
MEMMAN		1176	1.3	1	0.01
LINK		1	0.0	0	0.00
DELINK		1	0.0	0	0.00
CHECKPOINT		320	0.0	0	0.00

.

.

.

The [Table 5-3](#) lists the statistics for the queues and what they indicate.

Table 5-3. TCP Statistics for QUEUE INFO

Statistic	TERMPPOOL and SERVERPOOL	MEMMAN	LINK	DELINK	CHECK- POINT
REQ CNT (request count)	Number of requests for buffer allocation	Number of requests for SCREEN COBOL code (cache misses)	Number of requests to establish a link	Number of requests for dissolving a link	Number of requests for a checkpoint
% WAIT	Percentage of requests for buffer allocation that waited in the queue	Percentage of requests for SCREEN COBOL code that waited in the queue	Percentage of requests for a link that waited in the queue	Percentage of requests for dissolving a link that waited in the queue	Percentage of requests for a checkpoint that waited in the queue
MAX WAITS	Greatest number of requests for buffer allocation on the queue at one time	Greatest number of requests for SCREEN COBOL code on the queue at one time	Greatest number of requests for a link on the queue at one time	Greatest number of requests for dissolving a link on the queue at one time	Greatest number of requests for a checkpoint on the queue at one time
AVG WAITS	Average number of requests for buffer allocation on the queue at one time	Average number of requests for SCREEN COBOL code on the queue at one time	Average number of requests for a link on the queue at one time	Average number of requests for dissolving a link on the queue at one time	Average number of requests for a checkpoint on the queue at one time

REQ CNT

The following subsections describe the entries for REQ CNT. (Note that requests waiting on the \$RECEIVE queue of a server process are not collected for statistical purposes.)

TERMPOOL and SERVERPOOL

For TERMPOOL and SERVERPOOL, REQ CNT indicates the number of requests for buffer allocation that were queued, and implies the number of SEND statements.

MEMMAN

For MEMMAN, REQ CNT indicates the number of requests for SCREEN COBOL code that were queued. This number should be small after startup.

When a terminal task is about to execute, it asks the TCP to fetch all its defined segments. This counts as a single request in the REQ CNT entry for MEMMAN.

LINK

For LINK, REQ CNT indicates the number of requests for a link that were queued.

For each SEND statement that finds no server available, the number for REQ CNT is incremented by 1.

REQ CNT does not indicate the number of SEND commands executed by the TCP or how often link requests are sent to the PATHMON process (because the task on the LINK queue might be taken off that queue before the TCP relays its request to the PATHMON process).

You get better performance if you tune your system to make %WAIT zero or less.

DELINK

For DELINK, REQ CNT indicates the number of times a dynamic server was idle and the DELETEDELAY timer was started.

REQ CNT does not indicate how often the DELINK requests are sent to the PATHMON process because the request might be taken off the DELINK queue before the DELETEDELAY timer expires.

CHECKPOINT

For CHECKPOINT, REQ CNT indicates the number of requests for a checkpoint. (All checkpoints join the queue whether the checkpointer task is busy or not.)

% WAIT

The following subsections describe the entries for %WAIT.

TERMPOOL and SERVERPOOL

For TERMPOOL and SERVERPOOL, % WAIT indicates the percentage of requests for buffer space that waited in the queue.

A nonzero value indicates that some task had to wait until sufficient space was available. For information about sizing TERMPOOL and TERMBUF, see [POOL INFO](#) on page 5-8 in this section.

LINK

For LINK, %WAIT indicates the percentage of requests for a link that waited in the queue:

- A nonzero value indicates that some of the transactions had to wait for a link to become available.

This number, however, does not indicate the count of requests for a link to the PATHMON process. The number of requests for links will be less than or equal to the %WAIT number.

- If there are insufficient links available, the PATHMON process generates a LINK DENIED message to the log file (for the first occurrence of this problem).

The ability of TCPs and server processes to get links can profoundly affect system performance and response time. Frequently, link availability is determined by the attributes set for objects such as TCPs and server classes when a system is configured. For information about how you can configure your PATHMON environment for optimum link management and availability, see the *TS/MP System Management Manual*.

DELINK

For DELINK, % WAIT indicates the percentage of requests for dissolving a link that waited in the queue.

CHECKPOINT

For CHECKPOINT, %WAIT indicates the percentage of requests for a checkpoint that waited in the queue.

Terminal Statistics

Terminal statistics show the amount of TCP resources used by a terminal task. These statistics provide information about the following:

- Input and output operations (I/O INFO)
- Memory segments (AREA INFO)
- Response time (RESPONSE TIME INFO)

- Frequency distribution (optional)

[Example 5-5](#) shows a sample set of terminal statistics. Because frequency distribution is an optional item, this figure does not include distribution statistics. For an example of terminal statistics showing frequency distribution, see [Frequency Distribution](#) on page 5-25.

Example 5-5. Sample Terminal Statistics

```

TERM TERM1                                18 FEB 1996, 09:11:51
I/O INFO:      REQ CNT    MAX TSIZE    AVG TSIZE    I/O CNT
  DISPLAY              38         161         73         19
  ACCEPT              19          20          20         19
  SEND               19           2           2         19
  REPLY               **           2           2          **
  CHECKPOINT          **          910         469          **
AREA INFO:      MAX SIZE    AVG SIZE    CUR SIZE
  DATA             1106
  CODE              1928          992         1928
RESPONSE TIME INFO (TIME VALUES IN SECS):
  ACCEPT/SEND MESSAGE
    SUMMARY  # MEAS    AVG RESP    MAX RESP    MIN RESP    STAND DEV
              18        1.03        1.58        0.24        0.21

```

The TCP does not generate values for these fields: REPLY REQ CNT, REPLY I/O CNT, CHECKPOINT REQ CNT, CHECKPOINT I/O CNT, and DATA AVG SIZE.

I/O Info

The I/O INFO section in the statistics display provides information about the I/O operations requested by a terminal task.

These operations are requested by using these SCREEN COBOL statements:

- DISPLAY
- ACCEPT
- SEND
- REPLY
- CHECKPOINT

[Example 5-6](#) on page 5-19 shows only the I/O INFO section of the sample terminal statistics.

Example 5-6. Sample Terminal Statistics for I/O INFO

```

TERM TERM1                                18 FEB 1996, 09:11:51
I/O INFO:      REQ CNT    MAX TSIZE    AVG TSIZE    I/O CNT
  DISPLAY              38          161          73          19
  ACCEPT              19           20          20          19
  SEND               19            2            2          19
  REPLY              **            2            2          **
  CHECKPOINT         **           910         469          **
.
.
.

```

The [Table 5-4](#) lists the statistics for the I/O operations and what they indicate.

Table 5-4. Terminal Statistics for I/O INFO

Statistic	DISPLAY	ACCEPT	SEND	REPLY	CHECK- POINT
REQ CNT	Number of times the TCP executes these statements: DISPLAY BASE, DISPLAY DYNAMIC, DISPLAY OVERLAY, DISPLAY, SEND MESSAGE, RESET, TURN, CLEAR INPUT, and SCROLL	Number of times the TCP executes an ACCEPT statement	Number of time the TCP executes a SEND statement	N.A.	N.A.

Table 5-4. Terminal Statistics for I/O INFO

Statistic	DISPLAY	ACCEPT	SEND	REPLY	CHECK-POINT
MAX TSIZE (transfer size)	Size, in bytes, of the largest amount of data transferred to the terminal	Size, in bytes, of the largest amount of data transferred from the terminal	Size, in bytes, of the largest amount of data transferred to a server	Size, in bytes, of the largest amount of data transferred from a server	Size, in bytes, of the largest amount of data transferred for a checkpoint request
AVG TSIZE	Size, in bytes, of the average amount of data transferred to the terminal	Size, in bytes, of the average amount of data transferred to the terminal	Size, in bytes, of the average amount of data transferred to a server	Size, in bytes, of the average amount of data transferred from a server	Size, in bytes, of the average amount of data transferred for a checkpoint request
I/O CNT	Number of I/O operations caused by the DISPLAY statements	Number of I/O operations caused by the ACCEPT statements	Number of I/O operations caused by the SEND statements (For SEND, I/O CNT is always the same as REQ CNT.)	N.A.	N.A.

REQ CNT

REQ CNT indicates the number of times that the TCP executes the SCREEN COBOL DISPLAY, ACCEPT, or SEND statements.

Statistics are not provided for REPLY statements because there are no explicit requests for REPLY. Statistics are not provided for CHECKPOINT operations because the TCP performs more checkpoint operations than there are SCREEN COBOL checkpoint requests.

DISPLAY Statements

Because the DISPLAY statements (DISPLAY BASE, DISPLAY DYNAMIC, DISPLAY OVERLAY, DISPLAY, RESET, TURN, CLEAR INPUT, and SCROLL) do not always

require a physical I/O operation, REQ CNT indicates the number of logical I/O operations—not the number of physical I/O operations.

The DISPLAY statements generally cause the TCP to generate data for the terminal and place the data in a TERMPOOL buffer, but do not always require that the TCP write the buffer to the terminal. Consequently, a logical I/O operation does not necessarily generate a physical I/O operation.

When the TCP receives a request to write the buffer to a terminal, DISPLAY I/O CNT is incremented by 1.

In general, the number of logical I/O operations (indicated by REQ CNT) should be greater than the number of physical I/O operations (indicated by I/O CNT). (This is not true for IDS, however.) If the number of logical I/O operations is not greater, you should check for the following:

- The buffer might be too small.
- The TCP requester might be coded with a delay or other statements that cause the TCP to perform two-part I/O operations.

If more than one I/O is required, you might want to examine the requester code.

MAX TSIZE

MAX TSIZE indicates the size of the largest data transfer for the DISPLAY, ACCEPT, SEND, and REPLY statements and for checkpoint operations. To set MAX TSIZE, ask your application developers for the value of the largest amount of data that could be transferred from a server.

DISPLAY Statements

For DISPLAY, MAX TSIZE indicates the size of the largest amount of data sent to the terminal.

If the number indicated for MAX TSIZE is greater than that specified for TERMBUF, multiple I/O operations are required to display the data. To reduce the number of I/O operations required to display the data, you should:

- Increase the size of TERMBUF (using the SET TCP TERMBUF command) to MAXTSIZE +8.
- Increase the size of TERMPOOL (using the SET TCP TERMPOOL command) by the corresponding amount.

REPLY

For REPLY, MAXTSIZE indicates the largest amount of data transferred from a server.

To determine the most efficient value for the SET TCP MAXREPLY command, use the value indicated for MAXTSIZE.

AVG TSIZE

AVG TSIZE indicates the size of the average data transfer for the DISPLAY, ACCEPT, SEND, and REPLY statements and for checkpoint operations.

DISPLAY Statements

For DISPLAY, AVG TSIZE indicates the size of the average buffer sent to the terminal.

If the number indicated for AVG TSIZE is greater than that specified for TERMBUF, multiple I/O operations are required to display the data. To reduce the number of I/O operations required to display the data, you should:

- Increase the size of TERMBUF (using the SET TCP TERMBUF command) to MAXTSIZE +8.
- Increase the size of TERMPOOL (using the SET TCP TERMPOOL command) by the corresponding amount.

I/O CNT

I/O CNT indicates the number of physical I/O operations (when the TCP actually writes data to the terminal or sends data to a server) generated by the SCREEN COBOL DISPLAY, ACCEPT, or SEND statements.

Statistics are not provided for REPLY statements because there are no explicit requests for REPLY. Statistics are not provided for CHECKPOINT operations because the TCP performs more checkpoint operations than there are SCREEN COBOL checkpoint requests.

Block-Mode Terminals

For terminals running in block mode, the TCP writes data to the terminal when:

- The buffer is full.
- An ACCEPT statement is executing.

For some block-mode terminals (6500-series terminals and 3270 terminals in SNAX and AM configurations), the ACCEPT statement generates two I/O operations: the first I/O reads the key; the second I/O reads the data.

- Any of the DISPLAY statements are followed by certain other SCREEN COBOL statements, such as DELAY.

You can specify the size of the output buffer allocated from TERMPOOL with the SET TCP TERMBUF command. If the buffer is large enough, the TCP need not write data to the terminal simply because the buffer is full.

The minimum DISPLAY I/O CNT for block-mode terminals is equal to ACCEPT REQ CNT.

Conversational-Mode Terminals

For terminals running in conversational mode, the TCP writes output to the terminal whenever it executes a DISPLAY statement that generates data for a screen line.

You cannot specify the size of the terminal buffer for conversational-mode terminals.

Area Info

The AREA INFO section of the statistics display provides information about the data and code segments used by a terminal task.

[Example 5-7](#) shows only the AREA INFO section of the sample terminal statistics.

Example 5-7. Sample Terminal Statistics for AREA INFO

```

TERM TERM1                                18 FEB 1996, 09:11:51
.
.
.
AREA INFO:      MAX SIZE    AVG SIZE    CUR SIZE
  DATA          1106             910
  CODE          1928            1928
.
.
.

```

The [Table 5-5](#) lists the statistics for the memory segments and what they indicate.

Table 5-5. Terminal Statistics for AREA INFO

Statistic	For CODE, Indicates:	For DATA, Indicates:
MAX SIZE	Size, in bytes, of the largest single terminal request for code space (This number is the sum of all the code segment sizes in the request.)	Size, in bytes, of the largest amount of data space requested for terminal context
AVG SIZE	Size, in bytes, of the average terminal request for code space	N.A.
CUR SIZE	Size, in bytes, of the latest terminal request for code space	Size, in bytes, of the current amount of data space requested for terminal context

The STATS TERM display includes the value for Slot 0 only.

MAX SIZE

By comparing the MAX SIZE values for all terminals controlled by the TCP, you can determine the largest MAX SIZE value.

You can use this value to determine the most efficient value for the SET TCP MAXTERMDATA command.

Response Time Info

The RESPONSE TIME INFO section of the statistics display provides information about the response time of terminal tasks. Response time for a terminal is the length of time that elapses from the moment a user presses a function key until the next opportunity to press a function key.

[Example 5-8](#) shows only the RESPONSE TIME INFO section of the sample terminal statistics.

Example 5-8. Sample Terminal Statistics for RESPONSE TIME INFO

```

TERM TERM1                                18 FEB 1996, 09:11:51
.
.
.
RESPONSE TIME INFO (TIME VALUES IN SECS) :
ACCEPT/SEND MESSAGE
SUMMARY  # MEAS      AVG RESP      MAX RESP      MIN RESP      STAND DEV
          18          1.03          1.58          0.24          0.21
.
.
.

```

The [Table 5-6](#) lists the response time statistics and what they indicate.

Table 5-6. Terminal Statistics for Response Time

Statistic	Indicates:
# MEAS	Total number of collected measurements
AVG RESP	Average response time calculated from the collected measurements
MAX RESP	Maximum response time recorded during the given time interval (since statistics were turned on or reset)
MIN RESP	Minimum response time recorded during the given time interval (since statistics were turned on or reset)
STAND DEV	Standard deviation calculated from the collected measurements

The following are the limits imposed on measurement collection:

- Measurements will stop for any terminal upon the 99,999,999th measurement.
- Any measurement greater than 99,999,999.99 seconds is discarded.
- Once a measurement counter reaches 99,999,999.99 seconds, measurements will stop being collected for this terminal.
- Response times are rounded to the nearest hundredth of a second by the PATHMON process.

Frequency Distribution

When you specify the FREQTABLE option of the STATS TERM command, the PATHMON process displays supplemental statistics on response time within a given time interval.

The supplemental information is contained in the frequency distribution table, as shown in [Example 5-9](#). A separate frequency distribution table is generated for each terminal.

Example 5-9. Sample Terminal Statistics With Frequency Distribution Table

```

TERM TERM1
I/O INFO:          REQ CNT    MAX TSIZE    AVG TSIZE    I/O CNT
  DISPLAY          4568         161          69         2284
  ACCEPT           2284          20          20         2284
  SEND             2284          2           2         2284
  REPLY            2284          2           2
  CHECKPOINT              910         465
AREA INFO:      MAX SIZE    AVG SIZE    CUR SIZE
  DATA          1106              910
  CODE           1928          964      1928
RESPONSE TIME INFO (TIME VALUES IN SECS) :
ACCEPT/SEND MESSAGE
SUMMARY # MEAS    AVG RESP    MAX RESP    MIN RESP    STAND DEV
        2283      2.04      4.60      0.68      0.50
FREQUENCY DISTRIBUTION:
TIME INTERVAL (0.13 SECS)    # MEAS    CUM %
      TI01 <      0.80      1      0.0
0.80 <= TI02 <      0.94      1      0.0
0.94 <= TI03 <      1.08     97     4.3
1.08 <= TI04 <      1.22     99     8.6
1.22 <= TI05 <      1.35     70    11.7
1.35 <= TI06 <      1.49    109    16.5
1.49 <= TI07 <      1.63     42    18.3
1.63 <= TI08 <      1.76     77    21.7
1.76 <= TI09 <      1.90    156    28.5
1.90 <= TI10 <      2.04    484    49.7
2.04 <= TI11 <      2.18    487    71.0
2.18 <= TI12 <      2.31    166    78.3
2.31 <= TI13 <      2.45     80    81.8
2.45 <= TI14 <      2.59     35    83.3
2.59 <= TI15 <      2.73    101    87.8
2.73 <= TI16 <      2.86     73    91.0
2.86 <= TI17 <      3.00    111    95.8
3.00 <= TI18 <      3.14     87    99.6
3.14 <= TI19 <      3.28      3    99.8
3.28 <= TI20      4    100.0

```

The TCP does not generate values for these fields: REPLY REQ CNT, REPLY I/O CNT, CHECKPOINT REQ CNT, CHECKPOINT I/O CNT, and DATA AVG SIZE.

Use the frequency distribution table to determine the number of transactions in progress. Knowing the number of transactions enables you to identify the number of links required, and thus the configuration you need.

The [Table 5-7](#) on page 5-26 lists the terminal statistics for frequency distribution shown in [Example 5-9](#) on page 5-25.

Table 5-7. Terminal Statistics for Frequency Distribution

Statistic	Indicates
TIME INTERVAL	Value of the time increment from one interval to the next
# MEAS	Total number of measurements collected during the given time interval
CUM %	Cumulative percentage of the total number of measurements the line item represents

The frequency distribution table is not generated if either of these conditions exists:

- There are fewer than 50 sample measurements collected at the time of the STATS request. In this case, the following text appears at the bottom of the STATS TERM display:

```
FREQUENCY DISTRIBUTION:
  NOT PRODUCED:  LESS THAN 50 SAMPLE MEASUREMENTS TAKEN
```

- At the fiftieth measurement, the time increment calculated is less than 0.01 second. In this case, the following text appears at the bottom of the STATS TERM display:

```
FREQUENCY DISTRIBUTION:
  NOT PRODUCED:  COMPUTED INTERVAL LESS THAN .01 SECS
```

Examples of System Management Tasks

Example Task Overview

This section presents an example that demonstrates typical system management operations. The example illustrates configuration guidelines and steps, using many of the commands described throughout this manual and the *TS/MP System Management Manual*. By following the steps in the example, you can create and configure a PATHMON environment that runs a small sample application.

The example includes three general tasks:

- Configuring and starting the PATHMON process and its objects
- Restarting the environment and adding dedicated devices

Communicating with another PATHMON environment

Note. The example in this section includes some PATHCOM commands that apply to TS/MP and others that are specific to the HP NonStop Pathway/iTS product. For general descriptions and detailed syntax of the TS/MP commands, see the *TS/MP System Management Manual*.

Configuring and Starting a Simple PATHMON Environment (Task 1)

In this task, you configure and start a simple PATHMON environment that runs a small application. This application consists of a single screen program that communicates with a single server program. The application involves no database or other major component: its purpose is to demonstrate PATHMON environment management techniques rather than application design concepts.

The PATHMON environment requires a TCP to run the screen program, a PROGRAM object to allow users to run the application from temporary terminals, and a SERVER process to run the server program. In configuring and using this system, you demonstrate how:

- A new PATHMON environment is started (through a COLD start operation)
- A Pathway application is started from a temporary (PROGRAM) terminal (through a RUN command)
- A request is transmitted from a TCP to a server process (through the SEND statement in a screen program)

For purposes of this example, the sample source code for the screen and server programs appears in [Appendix G, Source Code for Programs in Section 6](#). Both the

screen and server programs are intended as brief samples that are easy to use: brevity and simplicity are emphasized more than standard design and coding practices. (The screen program is written for terminals running in block mode. However, this program could easily have been written for terminals operating in conversational mode.)

Steps 1 through 4 describe how to code and compile your source programs for the application. Steps 5 through 11 describe how to configure, start, and stop your PATHMON environment.

1. Code the screen program for the application. To do this, use your text editor to create a source file named SC and enter the SCREEN COBOL code listed in [Example G-1](#) on page G-2.

2. Compile the screen program using the SCREEN COBOL compiler, as follows:

```
2> SCOBOLX /IN SC/
```

The compiler writes the object code, by default, to the files named POBJCOD and POBJDIR in your current volume and subvolume.

3. Code the server program for the application. To do this, use your text editor to create a source file named SV and enter the COBOL code listed in [Example G-2](#) on page G-5.

4. Compile the server program using the COBOL85 compiler, directing the object code output to a file named SVOBJ:

```
4> COBOL85 /IN SV/ SVOBJ
```

5. Using your text editor, create a file named COLD1 and enter the TACL commands shown in [Example 6-1](#):

Example 6-1. TACL Commands in the COLD1 File

```
PURGE LOG1
FUP CREATE LOG1,TYPE E, REC 132, EXT(100,100), MAXEXTENTS 100
PATHMON /NAME $AC1, CPU 1, OUT LOG1, NOWAIT/
PATHCOM /IN CONFIG1/ $AC1
PATHCOM $AC1
```

You use this COLD1 file as a command file to configure your PATHMON environment. The commands in the COLD1 file accomplish the following:

1. Create a log file named LOG1 for your PATHMON environment.
2. Create your PATHMON process, assign it the process name \$AC1, and run its primary process on CPU 1.
3. Start PATHCOM, which opens communication with your PATHMON process, reads PATHCOM commands from a command file named CONFIG1, and returns control to TACL.
4. Start PATHCOM again, so that you are prompted for a new PATHCOM command.

- Using your text editor, create the file named CONFIG1 to serve as the PATHCOM command file, and enter the PATHCOM commands shown in [Example 6-2](#):

Example 6-2. PATHCOM Commands in the CONFIG1 File

```
SET PATHMON BACKUPCPU 0

SET PATHWAY NODEINDEPENDENT ON
SET PATHWAY MAXASSIGNS 5
SET PATHWAY MAXEXTERNALTCPS 1
SET PATHWAY MAXPARAMS 5
SET PATHWAY MAXPATHCOMS 5
SET PATHWAY MAXPROGRAMS 2
SET PATHWAY MAXSERVERCLASSES 5
SET PATHWAY MAXSERVERPROCESSES 5
SET PATHWAY MAXSTARTUPS 5
SET PATHWAY MAXTCPS 2
SET PATHWAY MAXTERMS 5

START PATHWAY COLD!

SET TCP CPUS 1:2
SET TCP MAXSERVERCLASSES 5
SET TCP MAXSERVERPROCESSES 5
SET TCP MAXTERMS 5
SET TCP TCLPROG POBJ
ADD TCP EXAMPLE-TCP

SET PROGRAM TCP EXAMPLE-TCP
SET PROGRAM TMF OFF
SET PROGRAM TYPE T16-6520 (INITIAL EXAMPLE-SCREEN-PROGRAM)
ADD PROGRAM SIMPLE

SET SERVER CPUS 0:1
SET SERVER MAXLINKS 5
SET SERVER MAXSERVERS 3
SET SERVER NUMSTATIC 1
SET SERVER PROGRAM SVOBJ
ADD SERVER EXAMPLE-SERVER
```

The commands in the CONFIG1 file perform the following:

- Configure the backup CPU for the PATHMON process.
- Configure global limits (MAXPROGRAMS, MAXTCPS, and so forth).
- Start your PATHMON environment (with the COLD! option).
- Configure (define and add) the PATHMON-controlled objects for your system: TCP, PROGRAM, and SERVER.

In the CONFIG1 file, notice that:

- The SET PATHWAY MAXPATHCOMS command allows up to five PATHCOMs to run concurrently. The SET PATHWAY MAXTERMS command

allows up to five input-output devices to be used at the same time. These commands enable the PATHMON process and the TCP to support up to five application terminals simultaneously.

- The SET SERVER MAXLINKS command allows up to five concurrent links to exist between a server process and a TCP. This enables the server process to queue up to five requests, each of which is handled by the server process as soon as the current request is completed. (This queueing of multiple requests is also supported by the RECEIVE-CONTROL paragraph in the Environment Division of the server program.)
- The ADD PROGRAM command assigns the name SIMPLE to the PROGRAM object.

7. Issue an OBEY command to execute the commands in the COLD1 file:

```
8> OBEY COLD1
```

In response to the fourth command in the COLD1 file, the commands in the CONFIG1 file are also executed.

8. When you receive a prompt from PATHCOM, issue a RUN command to run the application from your command terminal, referencing the PROGRAM name SIMPLE:

```
= RUN SIMPLE
```

At this point, you are using your terminal as an application user at an application device, rather than as a system manager. If other temporary (PROGRAM) terminals are available, try running the application from those terminals as well. You can run the application from up to five terminals concurrently.

When the application begins execution, it displays a screen that contains the text MESSAGE NUMBER as a prompt for you to enter a digit: 0, 1, or 2.

Select a digit and press the F1 function key. The screen program, through the TCP, sends a message containing the digit to the server. The server replies according to the digit it receives, directing the TCP's screen program to display a message that indicates the digit. For instance, if you enter 2, the screen program displays:

```
THIS IS MESSAGE NUMBER TWO
```

You can repeat the digit entry and transmission as often as you like. If you enter a character other than 0, 1, or 2, or press the wrong function key, the screen program displays an error message.

You can terminate the application at any time by pressing the F16 function key; control returns to PATHCOM.

9. Shut down your PATHMON environment by entering:

```
= SHUTDOWN2, MODE ORDERLY
```

In response, the PATHMON process shuts down the PATHMON-controlled objects and then terminates itself.

10. Terminate your interaction with PATHCOM by entering:

```
= EXIT
9>
```

PATHCOM returns control to TACL.

11. You can examine the log file to see a record of the events that took place while your PATHMON environment was running. To display the log file, enter:

```
10> FUP COPY LOG1, , FOLD, RECOUT 79, SHARE
```

The display appears in the format shown in [Example 6-3](#). Note that [Example 6-3](#) shows the format only. The figure does not show the actual PATHMON log file from Task 1.

Example 6-3. Sample LOG1 File

```
03NOV94, 11:43 $AC1:PATHMON - T8344D30- (31OCT94)
03NOV94, 11:43 $AC1:STATUS - *1043* TCP EXAMPLE-TCP, STARTED
03NOV94, 11:43 $AC1:STATUS - *1043* TERM 077-AL-000, STARTED
03NOV94, 11:44 $AC1:STATUS - *1047* TERM 007-AL-000, STOPPED: *3052 TERMINAL
STOPPED BY PROGRAM --- INST CODE: EXIT AT OFFSET %54 IN PROGRAM UNIT EXAMPLE-
SCREEN-PROGRAM (9) IN TCLPROG \SYS1.$SKULLF.RTLC10X.POBJ
03NOV94, 11:44 $AC1:STATUS -*1047* TCP EXAMPLE-TCP, STOPPED
03NOV94, 11:44 $AC1:STATUS -*1150* PATHMON STARTING SHUTDOWN - ORDERLY
03NOV94, 11:44 $AC1:STATUS -*1005* SHUTDOWN
03NOV94, 11:45 $AC1:STATUS -*1028* LOG1 FILE OPENED
03NOV94, 11:46 $AC1:STATUS -*1028* LOG1 FILE OPENED
03NOV94, 11:47 $AC1:STATUS -*1043* TCP EXAMPLE-TCP,STARTED
03NOV94, 11:47 $AC1:STATUS -*1043* TERM SAMPLE-TERM,STARTED
03NOV94, 11:47 $AC1:STATUS -*1043* TERM TERM-1,STARTED
03NOV94, 11:48 $AC1:STATUS -*1047* TERM SAMPLE-TERM,STOPPED: *3052* TERMINAL
STOPPED BY PROGRAM--INST CODE: EXIT AT OFFSET %54 IN PROGRAM UNIT EXAMPLE-
SCREEN-PROGRAM (9) IN TCLPROG \SYS1.$SKULLF.RTC10X.POBJ
03NOV94, 11:51 $AC1:STATUS -*1047* TERM TERM-1,STOPPED
03NOV94, 11:51 $AC1:STATUS -*1047* TCP EXAMPLE-TCP,STOPPED
03NOV94, 11:51 $AC1:STATUS -*1150* PATHMON STARTING SHUTDOWN - ORDERLY
03NOV94, 11:52 $AC1:STATUS -*1005* SHUTDOWN
```

Restarting the System and Adding Configured TERM Objects (Task 2)

In this task, you restart the PATHMON environment (using the COOL start option) and then add configured TERM objects. In addition to your command terminal, this task requires at least one other terminal that you can assign as an application terminal.

Note. If you want, you can proceed to Task 3, “Communicating With Another PATHMON Environment,” without performing Task 2.

1. Choose a terminal that you would like to use as an application device (choose two or more terminals, if you wish) and do the following:

- a. Determine the file name by which the operating system recognizes each terminal. To do this, enter this command at this terminal:

```
20> WHO
```

TACL responds by displaying the file name for the terminal, plus other operating environment information.

- b. Suspend the TACL process by entering this command at this terminal:

```
21> PAUSE
```

This command frees the terminal for dedication to your PATHMON environment.

△ **Caution.** Do *not* plan to use your command terminal as an application terminal. That is, do not plan to associate a configured TERM object with your command terminal.

2. Return to your command terminal.
3. With your text editor, create a file named COOL1 and enter the TACL commands shown in [Example 6-4](#):

Example 6-4. TACL Commands in the COOL1 File

```
PATHMON /NAME $AC1, CPU 1, NOWAIT/  
PATHCOM $AC1; START PATHWAY COOL!  
PATHCOM $AC1
```

You use this COOL1 file as a command file to restart your PATHMON environment with the COOL start option.

4. Issue an OBEY command to execute the commands in the COOL1 file:

```
22> OBEY COOL1
```

When the last command in the COOL1 file is executed, PATHCOM prompts you to enter a new command.

5. Use PATHCOM to interactively enter SET and ADD commands for defining and adding one or more TERM objects. For instance, to add two TERM objects named T1 and T2 that run under the TCP named EXAMPLE-TCP, you can enter commands such as:

```
= SET TERM FILE $AL  
= SET TERM INITIAL EXAMPLE-SCREEN-PROGRAM  
= SET TERM TCP EXAMPLE-TCP  
= SET TERM TMF OFF  
= ADD TERM T1  
= SET TERM FILE $JANE  
= ADD TERM T2
```


6. Enter the START commands for starting the TCP and TERM objects. Remember that you must start the TCP first, for example:

```
= START TCP EXAMPLE-TCP
= START TERM *, TCP EXAMPLE-TCP
```

When the START TERM command is executed, the application's initial screen appears on the terminals.

7. If you want to start the static server processes in the server class for the user, enter a START command:

```
= START SERVER EXAMPLE-SERVER
```

If you do not start the static server processes with the START command, the PATHMON process automatically starts a server process when the TCP requests a link to one.

8. At this point, you can use the application from any of the application terminals.

Communicating With Another PATHMON Environment (Task 3)

In this task, you create a second PATHMON environment and demonstrate how one PATHMON environment communicates with the other.

If you did not perform Task 2 of this example, you must first cool start the PATHMON environment that you defined in Task 1. To do this, perform Steps 3 and 4 of Task 2. Then, again perform Step 10 of Task 1 (exiting from PATHCOM).

To configure and start a second PATHMON environment, proceed as follows:

1. Using the File Utility Program (FUP), make a copy of the file SC that contains the screen program source code. Name this file SC2. To perform this step, enter:

```
4> FUP DUP SC, SC2
```

2. Using your text editor, access the source file SC2. Change the PROGRAM-ID paragraph to reflect the program name EXAMPLE-SCREEN-PROGRAM-REMOTE, as follows:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. EXAMPLE-SCREEN-PROGRAM-REMOTE.  <-- (Changed)
```

```
ENVIRONMENT DIVISION.
```

```

:
:
:
```

Next, change the SEND request to include the name of the PATHMON process that controls the first PATHMON environment:

```

:
:
```

```

      .
PROCEDURE DIVISION.
      .
      .
SEND-DATA.
MOVE 0 TO SEND-ERROR-FLAG
SEND ENTRY-MSG TO "EXAMPLE-SERVER"
UNDER PATHWAY "$AC1"                                <-- (Added)
REPLY CODE 0      YIELDS ENTRY-REPLY
      .
      .
      .

```

3. Compile the screen program in the SC2 file using the SCREEN COBOL compiler, as follows:

```
7> SCOBOLX /IN SC2/
```

At this point, two different program units exist in the POBJCOD and POBJDIR files: EXAMPLE-SCREEN-PROGRAM (from Task 1) and EXAMPLE-SCREEN-PROGRAM-REMOTE (from Task 3).

4. Using your text editor, create a file named COLD2 and enter the TACL commands shown in [Example 6-5](#).

Example 6-5. TACL Commands in the COLD2 File

```

PURGE LOG2
FUP CREATE LOG2,TYPE E, REC 132, EXT(100,100), MAXEXTENTS 100
ASSIGN PATHCTL, PATHCTL2
PATHMON /NAME $AC2, CPU 1, OUT LOG2, NOWAIT/
CLEAR ASSIGN PATHCTL
PATHCOM /IN CONFIG2/ $AC2
PATHCOM $AC2

```

You use the COLD2 file as a command file to configure the second PATHMON environment. The commands in this file:

- a. Create a log file named LOG2 for the PATHMON environment.
- b. Specify a PATHMON configuration file named PATHCTL2, whose name does not conflict with that of the PATHMON configuration file for the first PATHMON environment.
- c. Create a PATHMON process, assign it the process name \$AC2, and run its primary process on CPU 1.
- d. Start PATHCOM, which opens communication with the PATHMON process named \$AC2, reads PATHCOM commands from a command file named CONFIG2, and returns control to TACL.
- e. Start PATHCOM again, so that you are prompted for a new PATHCOM command.

- Using your text editor, create a file named CONFIG2 and enter the PATHCOM commands shown in [Example 6-6](#).

Example 6-6. PATHCOM Commands in the CONFIG2 File

```
SET PATHMON BACKUPCPU 0

SET PATHWAY MAXASSIGNS 0
SET PATHWAY MAXPARAMS 0
SET PATHWAY MAXPATHCOMS 5
SET PATHWAY MAXPROGRAMS 2
SET PATHWAY MAXSERVERCLASSES 0
SET PATHWAY MAXSERVERPROCESSES 0
SET PATHWAY MAXSTARTUPS 0
SET PATHWAY MAXTCPS 2
SET PATHWAY MAXTERMS 5

START PATHWAY COLD!

SET TCP CPUS 1:2
SET TCP MAXPATHWAYS 2
SET TCP MAXSERVERCLASSES 5
SET TCP MAXSERVERPROCESSES 5
SET TCP MAXTERMS 5
SET TCP TCLPROG POBJ
ADD TCP REMOTE-TCP

SET PROGRAM TCP REMOTE-TCP
SET PROGRAM TMF OFF
SET PROGRAM TYPE T16-6520 (INITIAL EXAMPLE-SCREEN-PROGRAM-REMOTE)
ADD PROGRAM REMOTE-PROGRAM
```

The commands in the CONFIG2 file configure the second PATHMON environment, start that system with the COLD start option, and configure a TCP and a PROGRAM for that system. With respect to your first PATHMON environment, the TCP is an external TCP. No SERVER objects are configured for the second PATHMON environment; your application, which spans both systems, uses the SERVER in the first PATHMON environment.

In the CONFIG2 file, note several commands that support communication between two or more PATHMON environments:

- The SET TCP MAXPATHWAYS command indicates that the TCP can communicate with up to two external PATHMON processes.
- The SET TCP MAXSERVERCLASSES command indicates that the TCP can communicate with up to five server classes simultaneously. (The server class running under the PATHMON process \$AC1 is used.)
- The SET TCP MAXSERVERPROCESSES command specifies that the TCP can establish links with up to five individual server processes, spread across all server classes, simultaneously. (The server processes running under the PATHMON process \$AC1 are used.)

If you examine the CONFIG1 file in [Example 6-2](#) for the first PATHMON environment, you will notice a SET PATHWAY MAXEXTERNALTCPS command.

This command specifies that the PATHMON process for the first PATHMON environment can communicate with one external TCP.

Returning to the CONFIG2 file, notice that the SET PROGRAM TYPE command specifies the same program name used as the PROGRAM-ID in Step 2.

6. Issue an OBEY command to execute the commands in the COLD2 file:

```
11> OBEY COLD2
```

In response to the sixth command in the COLD2 file, the commands in CONFIG2 are also executed. The commands in CONFIG2 configure and start your second PATHMON environment.

7. Issue a RUN command to run the application from your command terminal, referencing the PROGRAM name REMOTE-PROGRAM:

```
= RUN REMOTE-PROGRAM
```

You can use the application just as you did in Task 1. This time, however, you use a TCP running under the PATHMON process named \$AC2 to send requests to a server process running under the PATHMON process named \$AC1. Thus, from the viewpoint of your current user environment, \$AC2 is your local PATHMON process and \$AC1 is your remote PATHMON process.

At any time, you can terminate the application by pressing the F16 function key. Control returns to PATHCOM.

8. Shut down both PATHMON environments by entering:

```
= SHUTDOWN2, MODE ORDERLY
= OPEN $AC1
= SHUTDOWN2, MODE ORDERLY
```

9. Terminate your interaction with PATHCOM by entering:

```
= EXIT
10>
```

PATHCOM returns control to TACL.

10. To examine the log file for the second PATHMON environment, you can enter:

```
11> FUP COPY LOG2, , FOLD, RECOUT 79, SHARE
```

11. The display appears in the format shown in [Example 6-7](#).

Example 6-7. Logging Output in the LOG2 File

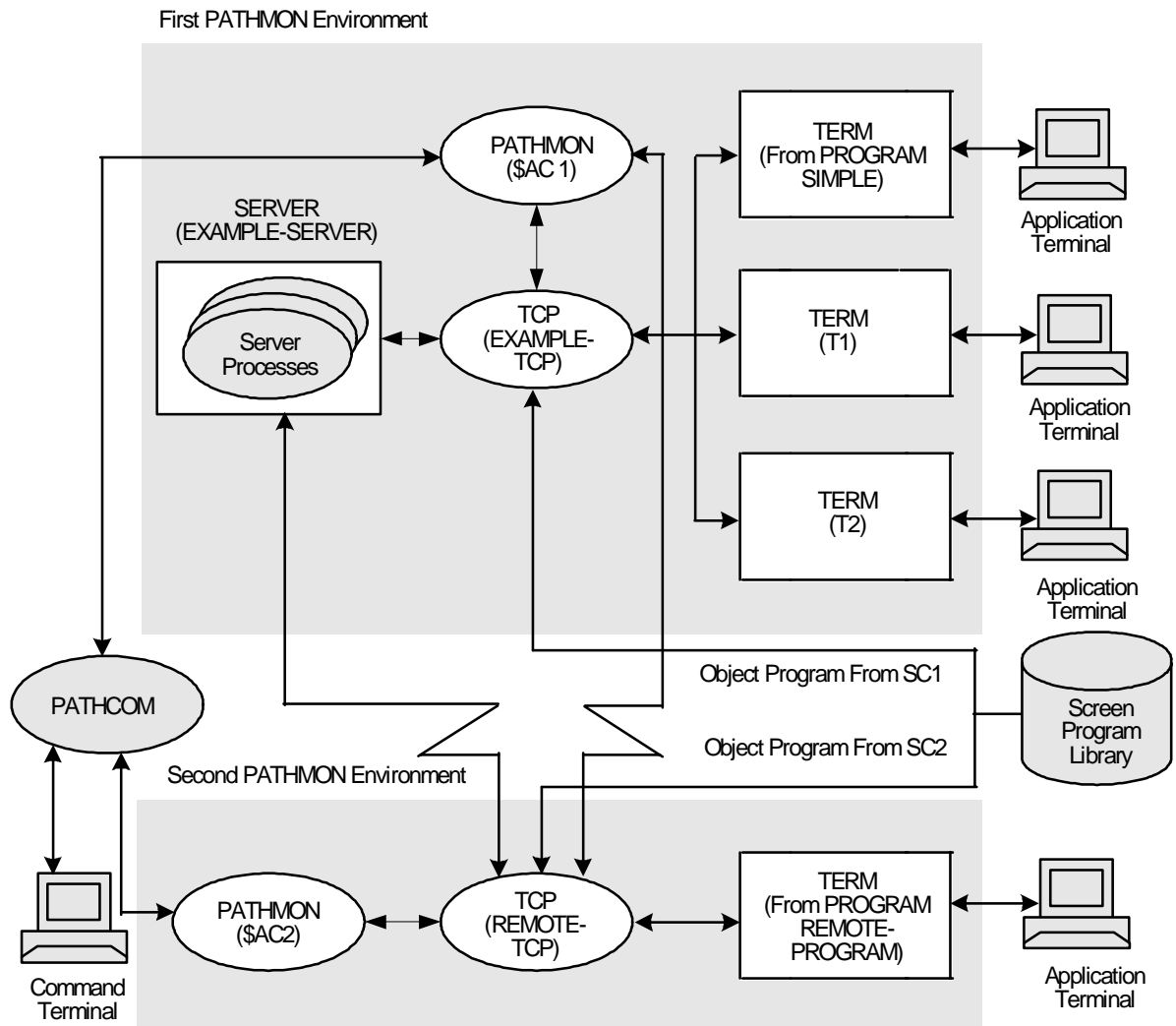
```
03NOV94, 11:50 $AC2:PATHMON - T8344D30 - (31OCT94)
03NOV94, 11:50 $AC2:STATUS - *1043* TCP REMOTE-TCP, STARTED
03NOV94, 11:50 $AC2:STATUS - *1043* TERM 077-AL-000, STARTED
03NOV94, 11:50: $AC2:STATUS - *1047* TERM 077-AL-000, STOPPED: *3052*
TERMINAL STOPPED BY PROGRAM---INST CODE: EXIT AT OFFSET %54 IN PROGRAM UNIT
EXAMPLE-SCREEN-PROGRAM-REMOTE (1) IN TCLPROG \SYS1.$SKULLF.RTLC10X.POBJ
03NOV94, 11:50 $AC2:STATUS - *1047* TCP REMOTE-TCP, STOPPED
03NOV94, 11:51 $AC1:STATUS -*1150* PATHMON STARTING SHUTDOWN - ORDERLY
03NOV94, 11:52 $AC1:STATUS -*1005* SHUTDOWN
```

This concludes Task 3 of the example.

Summary of Example

By running the complete example described in this section, you generate the PATHMON-controlled objects shown in [Figure 6-1](#).

Figure 6-1. PATHMON-Controlled Objects Produced by the Example



VST022.vsd

7 Overview of PATHCOM

This section contains overview information to help you get started using PATHCOM to manage HP NonStop Pathway/iTS objects in your PATHMON environment. See the *TS/MP System Management Manual* for more detailed information on how PATHCOM works, how to manage your PATHMON environment, and how to configure and manage the TS/MP objects.

PATHCOM Interface to PATHMON Environments

PATHCOM is a command language interpreter and interactive command interface to the PATHMON process. Using PATHCOM, which consists of sets of object-related commands, you can interactively define and manage all objects that can be controlled by the PATHMON process.

Some commands are processed directly by PATHCOM; others are processed by the PATHMON process, and one or more TCP or LINKMON processes might be involved in the execution of those commands. The commands described in this manual, which define, start, and stop objects controlled by the PATHMON process, are passed to the PATHMON process by PATHCOM.

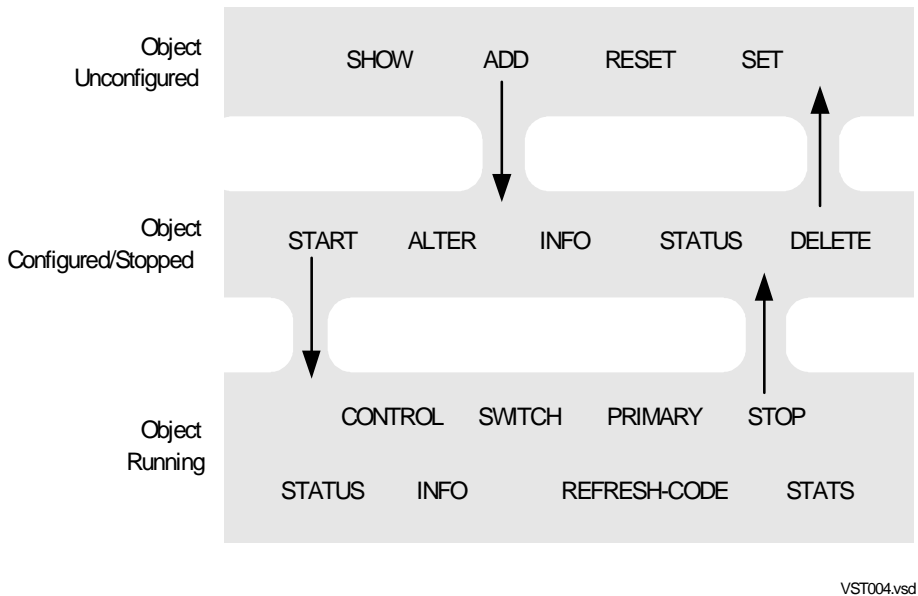
PATHMON-Controlled Objects

An object is an entity that you can see and control interactively by using PATHCOM or programmatically by using a management application that you write for the Pathway management programming interface. Each object type performs a specific function within a Pathway environment. The set of objects configured and controlled by the PATHMON process are referred to in this manual as the PATHMON environment or the PATHMON-controlled objects. The PATHMON-controlled objects that support requesters are the TCP, TERM, PROGRAM, and TELL objects, introduced in [Section 1, Introduction to Pathway/iTS System Management](#), and discussed throughout this manual. The PATHMON-controlled objects that support servers and the overall Pathway and PATHMON environments are the SERVER, PATHWAY, and PATHMON objects, respectively, also introduced briefly in Section 1. For a description of the SERVER, PATHWAY, and PATHMON objects, and a discussion of how requester and server objects interrelate, see the *TS/MP System Management Manual*.

Commands and Object States

The function of a PATHCOM command depends on the state of the object at the time you issue the command. For example, if an object such as a TCP is not configured, the START command for that object has no effect because that object cannot be started until it is configured using the ADD command. [Figure 7-1](#) on page 7-2 illustrates the relationship between the state of an object and the function of the PATHCOM commands.

Figure 7-1. PATHCOM Commands and Object States



Note. The CONTROL TERM commands must be issued when TERM is in the STOPPED state for COUPLE request.

The relationships illustrated in [Figure 7-1](#) apply for all of the PATHCOM commands and the PATHMON-controlled objects. Some PATHCOM commands affect only certain objects and their operation. For example, the ADD TERM command affects only configured TERM objects and the ALTER PROGRAM command affects only PROGRAM objects.

Note. The PATHMON process reports only destination object states; it does not report transitional states. To determine the current state of an object, use the appropriate STATUS command.

Command List

[Table 7-1](#) on page 7-3 is a list of the commands you can execute using PATHCOM to create and manage Pathway/iTS objects. The table entries provide the name of the command, its description, and the names of the objects influenced by the command. For PATHCOM commands that create and manage the TS/MP objects and control and operate PATHCOM itself, and for TACL commands that start the PATHMON and PATHCOM processes, see the *TS/MP System Management Manual*.

Table 7-1. PATHCOM Commands (page 1 of 2)

PATHCOM Commands	Description	Valid Objects *
ABORT	Immediately stops a terminal	TERM
ADD	Enters a description of an object into the configuration	PROGRAM, TCP, TERM
ALTER	Changes the attributes of a previously defined object while the object is stopped	PROGRAM, TCP, TERM
CONTROL**	Changes the attributes of an object while it is running	TCP and TERM
DELETE	Removes an object description from the system configuration	PROGRAM, TCP, TELL, TERM
INFO	Displays values for object attributes	PROGRAM, TCP, TELL, TERM
INSPECT	Causes the Pathway/iTS terminal to respond to HP <i>Inspect</i> symbolic debugger commands	TERM
PRIMARY	Causes primary and backup processes to resume executing in the CPUs defined in the object configuration	TCP
REFRESH-CODE	Causes the TCP to read the latest version of a called program unit	TCP
RESET	Resets object attributes to their default values	CMDCWD, PROGRAM, TCP, TERM
RESUME	Places suspended terminals into a running state	TERM
RUN	Initiates execution of a SCREEN COBOL program	PROGRAM
SET	Establishes values for object attributes before adding an object to the PATHMON environment	PROGRAM, TCP, TERM
SHOW	Displays current values for the object attributes and environmental settings for PATHCOM	CMDCWD, CMDVOL, PROGRAM, TCP, TERM
START	Starts one or more objects	TCP, TERM
STATS	Displays resource usage and system performance statistics	TCP, TERM
STATUS	Displays the current status of a PATHMON-controlled object	TCP, TERM
STOP	Stops a PATHMON-controlled object	TCP, TERM

Table 7-1. PATHCOM Commands (page 2 of 2)

PATHCOM Commands	Description	Valid Objects *
SUSPEND	Interrupts execution of a SCREEN COBOL program until a RESUME command is received	TERM
SWITCH	Causes primary and backup processes to exchange functions	TCP
TELL	Displays a message on a terminal	TERM

*Although TELL is listed here as a valid object, it is not like other objects. TELL is a temporary object that exists only for the duration of a tell message.

**The CONTROL TERM commands must be issued when TERM is in the STOPPED state for COUPLE request.

Command and Object Relationships

[Table 7-2](#) summarizes the relationship between the PATHCOM commands and the different PATHMON-controlled objects that support requesters.

Table 7-2. Commands and Objects (page 1 of 2)

PATHCOM Commands	Objects				
	TCP	External TCP	TERM	PROGRAM	TELL
ABORT			X		
ADD	X		X	X	
ALTER	X		X	X	
CONTROL	X		X		
DELETE	X		X	X	X
INFO	X		X	X	X
INSPECT			X		
PRIMARY	X				
REFRESH-CODE	X				
RESET	X		X	X	
RESUME			X		
RUN				X	
SET	X		X	X	
SHOW	X		X	X	
START	X		X		
STATS	X	X	X		
STATUS	X	X	X		

Table 7-2. Commands and Objects (page 2 of 2)

PATHCOM Commands	Objects				
	TCP	External TCP	TERM	PROGRAM	TELL
STOP	x		x		
SUSPEND			x		
SWITCH	x				
TELL			x		

Command Format

When entering PATHCOM commands, consider:

- In general, options within a command are position independent. That is, you can place the options in any order within the command line. For example, the following command:

```
SUSPEND TERM *, STATE RUNNING, TCP TCPA !
```

could also be entered as:

```
SUSPEND TERM *, ! TCP TCPA, STATE RUNNING
```

Exceptions to the independent positioning of options within a command are documented with the individual commands.

- You can embed comments within a line of commands or you can enter comments on separate lines. Enclose the comments in brackets if a PATHCOM command is on the same line. If there are no PATHCOM commands on the same line as a comment, you only need to enter a left bracket to cause PATHCOM to recognize a comment. For example:

```
[PATHCOM treats this line as a comment.]
```

```
[PATHCOM treats this line as a comment too.]
```

```
=SET TERM AUTORESTART 4 [Here is a comment.],TCP tcp-1
```

Note. You cannot embed PATHCOM comments in TACL scripts. In TACL scripts, information enclosed in brackets is interpreted as a function, not as a comment. See the *TACL Reference Manual* for information on including comments in TACL scripts.

- A PATHCOM command line spans input records if the last nonblank character is an ampersand (&). Multiple commands, separated by semicolons, can appear on the same line. If command parameters are repeated, PATHCOM uses the last value entered for a parameter.

Interactive Mode

PATHCOM functions in interactive mode when you enter commands from a terminal keyboard. PATHCOM prompts for a command by printing an equal sign (=). When you enter a command, PATHCOM executes the command and issues another prompt.

In the following example, the first command starts the PATHMON process, the second command starts the PATHCOM process, the third line shows the PATHCOM signon, and the fourth line shows the PATHCOM prompt:

```
TACL 8> PATHMON/NAME $PM1,CPU 0,NOWAIT/
```

```
TACL 9> PATHCOM $PM1
```

```
PATHCOM - T9153C31 - (08SEP92)
=
```

Note. For more information about the first and second commands in this example and about the PATHCOM signon message, see the *TS/MP System Management Manual*.

You can group two or more commands separated by semicolons (;). For example, these commands can be entered in two ways:

```
= command-1; command-2
```

or:

```
= command-1
```

```
= command-2
```

Pressing the terminal Break key during the execution of a PATHCOM command (as described in the *TS/MP System Management Manual*) cancels the command; control remains with PATHCOM and you can reenter the command. Pressing the Break key in all other instances, including after the PATHCOM prompt appears, returns the terminal to the TACL command interpreter without terminating PATHCOM. Enter “pause” at the TACL prompt to return the PATHCOM prompt (=).

Running PATHCOM interactively allows you to monitor the status of the objects and inform the Pathway/iTS terminal operators about changing conditions using tell messages.

Noninteractive Mode

PATHCOM functions in noninteractive mode when it reads commands from a command file.

In the following example, PATHCOM reads commands from a file named CMDFILE and lists them on the device \$\$.#LP. When it encounters an end-of-file command or an EXIT command, PATHCOM terminates.

```
3> PATHCOM/IN CMDFILE,OUT $$.#LP, CPU 1, NOWAIT/$PM2
```

You can use DEFINES to specify names for the files that PATHCOM uses directly as IN, OUT, and OBEY command files. In the following example, PATHCOM reads commands from the command file specified by `DEFINE =CMD-FILE ($DATA.PW.CONFIG)` and lists them on the device specified by `DEFINE =OUT-FILE ($S)`:

```
12> ADD DEFINE =CMD-FILE, CLASS MAP, FILE $DATA.PW.CONFIG
13> ADD DEFINE =OUT-FILE, CLASS SPOOL, LOC $S, REPORT
    "CONFIG"
14> PATHCOM /IN =CMD-FILE, OUT =OUT-FILE/ $PM
```

For detailed information about DEFINES in the *Guardian* operating environment, see the *TACL Reference Manual* and the *Guardian User's Guide*.

PATHMON Configuration File

The PATHMON process maintains a configuration file containing configuration information about the objects it controls. This file is usually called:

```
$default-volume.default-subvolume.PATHCTL
```

If a file named PATHCTL does not already exist, the PATHMON process automatically creates this file as part of the cold-start operation. The PATHCTL file is an Enscribe key-sequenced file and has a file code of 310. You can change the name of this file by using the TACL ASSIGN command and by referring to the logical file PATHCTL before running the PATHMON process. For more information about the PATHMON configuration file, including restrictions on the use of multiple configuration files, see the *TS/MP System Management Manual*.

Guardian File Names

Each disk file in the Guardian file-system is identified by a unique, symbolic file name, described in the following paragraphs.

File Name Format

The name, and therefore the location, of a Guardian disk file is determined in four parts:

- `\node`—Identifies a specific system within a network
- `$volume`—Identifies a physical disk pack mounted on a disk unit
- `subvolume`—Identifies a disk file as a member of a related set of files as defined by the user
- `file-identifier`—Identifies a particular file within the subvolume

File names of disk files are represented to programs on HP NonStop systems by these four parts concatenated into a contiguous string. Each part is separated from the other by a period as follows:

\node.\$volume.subvolume.file-identifier

When you supply only a partial file name as a command parameter, the internal representation of the file name is expanded into the full four-part file name. As a minimum, a partial file name must consist of the file identifier.

Each process and each device, such as a tape drive or printer, is identified the same way a disk file is identified. For example:

\TSB.\$TAPE1

specifies a particular tape drive on the system named \TSB. If a PATHMON environment is running on this system, only \$TAPE1 is required for the file name.

File Name Expansion

A command interpreter for a terminal has associated with it a default volume name and a default subvolume name. File name expansion is accomplished through the use of these default names. If you omit the volume name, the default volume name is used in its place.

Although the PATHMON process uses system names, it does not store TCP names in fully qualified network form. All other file names are kept in the form:

\node.\$volume.subvolume.file-identifier

PATHCOM uses the name that you supply in the CMDVOL command to expand a process file name before passing the process name to the PATHMON process.

Pathway Environment Control Commands

In general, the PATHCOM commands that control the PATHMON process and the Pathway environment as a whole deal with objects managed by the TS/MP product or with the global PATHWAY and PATHMON objects. These commands are described in detail in the *TS/MP System Management Manual*. One of these commands, however—the SET PATHWAY command—allows you to define attribute values for HP NonStop Pathway/iTS objects in addition to the TS/MP objects. Other commands can impact Pathway/iTS objects in various ways. This section describes those aspects of the SET PATHWAY command that affect TCP, TERM, PROGRAM, and TELL objects, and summarizes the effects of other commands upon these objects.

SET PATHWAY Command

Use the SET PATHWAY command to establish attribute values for the PATHWAY object. Use this command for configuring the PATHMON process before you start a PATHMON environment.

The following command description is limited to PATHWAY object attributes that are specific to Pathway/iTS objects, such as TCP, TERM, and PROGRAM objects. For information about using the SET PATHWAY command to establish values for PATHWAY object attributes that affect objects managed by TS/MP, or for global PATHMON and PATHWAY object attributes, see the *TS/MP System Management Manual*.

```
SET PATHWAY pw-attribute [ , pw-attribute ]...
```

pw-attribute is:

```
MAXTCPS number
MAXTERMS number
MAXEXTERNALTCPS number
MAXPROGRAMS number
MAXTELLQUEUE number
MAXTELLS number
MAXTMFRESTARTS number
```

pw-attribute

For information about configuration limits for environments, see [Appendix C, Configuration Limits and Defaults](#).

MAXTCPS number

is the maximum number of TCP objects that you can add to the PATHMON configuration file.

The total of the values for MAXTCPS, MAXEXTERNALTCPS, and MAXLINKMONS cannot exceed 800. A maximum of 255 TCPs, external TCPs, and LINKMON processes are allowed to hold links concurrently to a single server process.

The maximum number of concurrently running requester processes (TCPs, external TCPs, LINKMON processes, PATHCOM processes, and SPI requester processes) must not exceed 800.

number must be a value from 0 through 800. This is a required parameter; there is no default value.

MAXTERMS number

is the maximum number of TERM objects that can be added to the PATHMON configuration file. (Configured TERM objects are added by the ADD TERM command; temporary TERM objects are added by the RUN PROGRAM command.)

number must be a value from 0 through 4095. To enable the PATHMON process to accept an ALTER TERM command, specify a maximum of 4094 for this attribute.

This is a required parameter; there is no default value.

MAXEXTERNALTCPS number

is the maximum number of external TCPs that can communicate with the PATHMON process at the same time. An external TCP is one that is controlled by another PATHMON process and is, therefore, outside this PATHMON environment.

The total of the values for MAXTCPS, MAXEXTERNALTCPS, and MAXLINKMONS cannot exceed 800. A maximum of 255 TCPs, external TCPs, and LINKMON processes are allowed to concurrently hold links to a single server process. See [Appendix C, Configuration Limits and Defaults](#), for other requester process limits.

number must be a value from 0 through 800. If you omit this attribute, the value for MAXEXTERNALTCPS is 0; any attempt by an external TCP to access a server controlled by the PATHMON process fails.

MAXPROGRAMS number

is the maximum number of terminal template descriptions (PROGRAM objects) that you can add to the PATHMON configuration file.

number must be a value from 0 through 4095. If you omit this attribute, the default is 0; temporary TERM objects cannot be started in the PATHMON environment through the RUN PROGRAM command.

MAXTELLQUEUE *number*

is the maximum number of tell messages that can be queued for each terminal.

number must be a value from 0 through 300. If you omit this attribute, the default is 4.

MAXTELLS *number*

is the maximum number of tell messages that can exist concurrently within a PATHMON environment.

number must be a value from 0 through 4095. If you omit this attribute, the default is 32.

MAXTMFRESTARTS *number*

is the maximum number of times a TCP attempts to restart a logical TMF transaction after the transaction aborts and before the SEND error occurs. If this limit is reached and the SCREEN COBOL BEGIN-TRANSACTION verb does not have an ON ERROR clause, the terminal is suspended, but can be restarted.

number can be:

- 1 through 32,767 Specifies the number of restarts
- 0 Specifies no restarts
- 1 Specifies unlimited restarts

If you omit this attribute, the default is 5.

Consideration

All object attributes can be changed before a cold start.

Note. The SET PATHWAY command can also be used to configure attribute values for objects controlled through TS/MP, such as SERVER and LINKMON objects, in addition to attributes for the global PATHWAY and PATHMON objects. Parameters used for these objects include the following, all described in *TS/MP System Management Manual*: MAXASSIGNS, MAXPARAMS, MAXSERVERCLASSES, MAXSERVERPROCESSES, MAXSTARTUPS, MAXDEFINES, MAXLINKMONS, MAXPATHCOMS, MAXSPI, OWNER, and SECURITY.

Example

The following SET PATHWAY commands define requester-based elements in an example PATHMON environment:

```
SET PATHWAY MAXEXTERNALTCPS 5
SET PATHWAY MAXPROGRAMS 1
SET PATHWAY MAXTCPS 4
SET PATHWAY MAXTELLQUEUE 5
SET PATHWAY MAXTELLS 50
SET PATHWAY MAXTERMS 25
SET PATHWAY MAXTMFRESTARTS 5
```

Note. The configuration in this example does not include server-based objects that run under TS/MP, nor does it include the global PATHWAY and PATHMON objects.

Other Commands

Other Pathway environment control commands that can impact Pathway/iTS objects are as follows:

Command	Function or Effect on Pathway/iTS Objects
INFO PATHMON	Displays information about TCP, TERM, and PROGRAM objects
INFO PATHWAY	Displays information about TCP, TERM, and PROGRAM objects
LOG1 and LOG2	Specifies files to which TCP objects report errors and changes in object status
SHUTDOWN	Stops TCP objects
SHUTDOWN2	Stops TCP and TERM objects, and disables commands to add, alter, or delete TCP, TERM, and PROGRAM objects, and commands to delete TELL objects
START PATHWAY	Potentially affects definitions of TCP, TERM, and PROGRAM objects in the PATHMON configuration file

For information about these commands, see the *TS/MP System Management Manual*.

Terminal Control Process (TCP) Commands

This section describes the PATHCOM commands that define and control TCPs; the commands are listed in alphabetical order. The TCP commands are as follows:

To Perform the Following Tasks...	Use These PATHCOM Commands...
Define and remove TCPs	SET TCP ADD TCP CONTROL TCP DELETE TCP
Control TCPs	START TCP STOP TCP
Change TCP attributes	ALTER TCP RESET TCP
Obtain information about	INFO TCP SHOW TCP STATS TCP STATUS TCP
Perform other management tasks	SWITCH TCP PRIMARY TCP REFRESH-CODE TCP

This commands configure and control TCPs. With these commands you can do the following:

- Set and modify TCP attributes
- Add and delete TCPs
- Start and stop TCPs
- Display TCP attribute and status information
- Control TCP operation

Note. This manual documents only TCPs for which the code is in the program file \$SYSTEM.SYSTEM.PATHTCP2. All references to TCP denote this file name.

ADD TCP Command

Use the ADD TCP command to enter the name and description of a TCP into the PATHMON configuration file. Use this command after defining a TCP with the SET TCP command.

```
ADD TCP tcp-name [ , tcp-attribute ]...
```

tcp-name

specifies the name of a TCP. A TCP name can be from 1 through 15 alphanumeric or hyphen characters and must start with a letter, be unique within the PATHMON environment, and not be a PATHCOM reserved word.

tcp-attribute

specifies an attribute describing a TCP object. A TCP attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET TCP command. Use any of the attributes listed for the SET TCP command.

Consideration

To use the ADD TCP command, you must have a licensed copy of the HP NonStop Pathway/iTS product on your system; otherwise, the PATHMON process returns an error.

Examples

The following command adds the definition of the TCP named TCPAL to the PATHMON configuration file:

```
ADD TCP TCPAL
```

The following command adds the TCP named SALES-TCP to the PATHMON configuration file and changes the definition of the HOMETERM and SERVERPOOL attributes:

```
ADD TCP SALES-TCP, HOMETERM $CENTER, SERVERPOOL 25000
```

For guidelines on calculating appropriate configuration values, see the *TS/MP System Management Manual*.

ALTER TCP Command

Use the ALTER TCP command to change the attribute values of a TCP that was previously added to the PATHMON configuration file. The TCP must be stopped before you can alter an attribute value. For a list of the TCP attributes, see the [SET TCP Command](#) on page 9-18.

```
ALTER [ TCP ] tcp-name
```

```
{ , tcp-attribute }
{ , RESET tcp-keyword }
{ , RESET ( tcp-keyword [, tcp-keyword ]... ) }
```

tcp-keyword is:

AUTORESTART	INSPECT	POWERONRECOVERY
CHECK-DIRECTORY	MAXINPUTMSGLEN	PRI
CODEAREALEN	MAXINPUTMSG	PROCESS
CPUS	MAXPATHWAYS	PROGRAM
DEBUG	MAXREPLY	SERVERPOOL
DUMP	MAXSERVERCLASSES	SENDMSGTIMEOUT
GUARDIAN-LIB	MAXSERVERPROCESSES	STATS
GUARDIAN-SWAP	MAXTERMDATA	SWAP
HIGHPIN	MAXTERMS	TCLPROG
HOMETERM	NONSTOP	TERMBUF
		TERMPPOOL

tcp-name

specifies the name of a previously defined and added TCP.

tcp-attribute

specifies an attribute describing a TCP object. A TCP attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET TCP command. Use any of the attributes listed for the SET TCP command.

RESET

changes an existing attribute of a TCP to the default value.

If an attribute is not required and no default value exists, this option deletes the value set for the attribute.

If an attribute is required and no default value exists, the RESET option generates a syntax error and leaves the current attribute value unchanged.

tcp-keyword

specifies a single SET TCP command attribute keyword or several keywords separated by commas and enclosed in parentheses.

Consideration

If a TCP is configured with the TCP INSPECT attribute set to ON and a file name, the ALTER TCP RESET command sets the INSPECT attribute to OFF and deletes the file name.

Examples

The following commands change various TCP attribute values. The second and third commands reset attributes in addition to changing TCP attribute values:

```
ALTER TCP TCP-1,INSPECT ON (FILE $TERM1)
ALTER TCP-2,MAXTERMS 10,RESET PROGRAM
ALTER TCP TCP-3,PROGRAM $DATA.USER.PATHTCP2,RESET (PRI,TERMPPOOL)
```

CONTROL TCP Command

Use the CONTROL TCP command to change specific TCP attributes while the TCP is running and to record the changes in the PATHMON configuration file. If you restart the TCP after a CONTROL TCP command, the TCP runs with the attribute changes specified in this command.

If the TCP is not running when you issue this command, the PATHMON process returns an error.

```
CONTROL [ TCP ] { tcp-name
                  { ( tcp-name [ , tcp-name ]...)
                  { tcp-attribute [ , tcp-attribute ]... }
                  }

tcp-attribute is:

    BACKUPCPU number
    DUMP { ON [ ( FILE file-name ) ] | OFF }
    DUMPMEMORY { PRIMARY|BACKUP | BOTH } [(FILE file-name)]
    STATS { ON | OFF }
```

TCP *tcp-name*

specifies one or more TCPs. *tcp-name* can be either the name of a single TCP or several TCP names separated by commas and enclosed in parentheses.

BACKUPCPU *number*

specifies the CPU where the TCP backup process runs. The TCP primary process stops its existing backup process and creates a new backup in the CPU that you specify.

If the CPU you specify for the BACKUPCPU attribute is the CPU where the backup process currently resides, the TCP still stops its backup process and starts a new backup process in the same CPU.

DUMP { ON | OFF }

specifies whether the TCP performs a memory dump if it encounters an internal or fatal error.

If you omit this attribute, the default is OFF.

Note. The value defined for the CONTROL TCP DUMP attribute overrides the value defined in the SET TCP DUMP attribute.

ON

directs the TCP to create one or more disk files and write the contents of its data stack and its extended data segment to these files.

After creating 10 primary process and 10 backup process dump files, the TCP stops performing memory dumps. This prevents writing over existing dump files that may indicate the original cause of a problem, and prevents filling a disk with dump files.

FILE *file-name*

specifies the name of the file that the TCP creates for its dump operation. The format of the file name can be any of the following:

```
[ \node. ] $volume.subvolume
[ \node. ] $volume.subvolume.ZZ
[ \node. ] $volume.subvolume.filename
```

\node can be either a specific node name or ***, a generic name representing the node on which the PATHMON process is currently running.

For example, the following file name directs the TCP to create the appropriate ZZTCP_{nn}P or ZZTCP_{nn}B file on volume \$DATA using subvolume TCP05DMP:

```
(FILE $DATA.TCP05DMP)
(FILE $DATA.TCP05DMP.ZZ)
```

The following unique name for *file-name*, directs the primary TCP to name its dump file TCP05P and directs the backup TCP to name its dump file TCP05B:

```
(FILE $DATA.PWAYDMPS.TCP05)
```

Both the primary and backup dump files are created on volume \$DATA using subvolume PWAYDMPS. The P (primary TCP dump) and B (backup TCP dump) characters replace the last character of an eight-character name, or are appended to the end of a shorter dump file name. If the dump file already exists when the dump operation occurs, the TCPs reuse the existing file by purging the file's information and writing the latest memory dump.

If a file-system error interferes with creating a file that you specify, the TCP creates the dump file on the same subvolume as its program object file and uses the form ZZTCP_{nn}X for the file name.

If you do not specify *file-name*, the TCP creates its dump file on the same volume and subvolume as the PATHMON configuration file. The format of the file names are ZZTCP_{nn}P and ZZTCP_{nn}B, where *nn* is a number from 01 to 10, and P and B indicate the primary or the backup process, respectively, that the TCP dumped.

OFF

directs the TCP not to write data stack information to a file when it encounters an internal or fatal error.

DUMPMEMORY { PRIMARY | BACKUP | BOTH } [(FILE *file-name*)]

forces the TCP to write the contents of its data stack and its extended data segment to a file.

PRIMARY

specifies a memory dump of the primary TCP process only. Use this option unless a TCP is reporting errors from its backup process.

BACKUP

specifies a memory dump of the backup TCP process only. Use this option if a TCP is reporting errors from its backup process.

BOTH

specifies a memory dump of both the primary and the backup TCP process. Use this option if a TCP is reporting errors from its backup process and you also want a dump of the primary TCP.

FILE *file-name*

specifies the name of the file that the TCP creates for its dump operation. The format of the file name can be any of the following:

```
[ \node. ] $volume.subvolume
[ \node. ] $volume.subvolume.ZZ
[ \node. ] $volume.subvolume.filename
```


\node can be either a specific node name or ***, a generic name representing the node on which the PATHMON process is currently running.

For example, the following file name directs the TCP to create the appropriate ZZTCP_{nn}P or ZZTCP_{nn}B file on volume \$DATA using subvolume TCP05DMP:

```
(FILE $DATA.TCP05DMP)
(FILE $DATA.TCP05DMP.ZZ)
```

The following unique name for *file-name*, directs the primary TCP to name its dump file TCP05P and directs the backup TCP to name its dump file TCP05B:

```
(FILE $DATA.PWAYDMPS.TCP05)
```

Both the primary and backup dump files are created on volume \$DATA using subvolume PWAYDMPS. The P (primary TCP dump) and B (backup TCP dump) characters replace the last character of an eight-character name, or are appended to the end of a shorter dump file name. If the dump file already exists when the dump operation occurs, the TCPs reuse the existing file by purging the file's information and writing the latest memory dump.

If a file-system error interferes with creating a file that you specify, the TCP creates the dump file on the same subvolume as its program object file and uses the form ZZTCP_{nn}X for the file name.

If you do not specify *file-name*, the TCP creates its dump file on the same volume and subvolume as the PATHMON configuration file. The format of the file names are ZZTCP_{nn}P and ZZTCP_{nn}B, where *nn* is a number from 01 to 10, and P and B indicate the primary or the backup process, respectively, that the TCP dumped.

STATS { ON | OFF }

specifies whether to enable or disable the statistics collection mechanism within the TCP while the TCP is running.

If you omit this attribute, the default is OFF.

ON

resets all STATS counters to zero and then activates the statistics collection mechanism.

OFF

resets all STATS counters to zero and then deactivates the statistics collection mechanism.

Note. You must enter all STATS commands before issuing the CONTROL TCP command with the OFF attribute because this attribute resets all the statistical counters to zero.

Considerations

- If the TCP primary process is running in the CPU defined for the TCP backup process because of a SWITCH command or some other processor change, the BACKUPCPU attribute causes the PATHMON process to change the PATHMON configuration file so that it contains the current CPU numbers for both the primary and backup TCPs.
- If a TCP was configured with the NONSTOP attribute set to 0, changing the BACKUPCPU attribute only changes the backup CPU value in the PATHMON configuration file; no processor change occurs.
- When you cool start a PATHMON environment after using the CONTROL TCP command, the TCPs start according to the changes recorded in the PATHMON configuration file.
- If you request the help of your service provider in analyzing a problem, the representative will likely require a dump file. It is therefore recommended that you always set the DUMP option to ON for production systems.

The DUMPMEMORY option is not a substitute for setting the DUMP option to ON. DUMPMEMORY is primarily useful in a controlled troubleshooting situation where you need to take a snapshot of the TCP's internal state at a particular time -before it encounters a fatal error.

Errors

The following table lists the most common errors that can occur during the processing of the CONTROL TCP command:

This Message...	Is Displayed When...
1070 TCP BACKUP NOT UP	You have issued a CONTROL TCP, BACKUPCPU command or a SWITCH TCP command directly after issuing a PRIMARY TCP command and the TCP backup process has not had enough time to be restarted.
1093 BACKUP PROCESSOR DOWN	For a multiprocessor system, you specified a backup CPU that does not exist or is not operational. For a single processor system, you included the BACKUPCPU attribute.
1095 ILLEGAL CPU NUMBER	You specified for the BACKUPCPU attribute the same CPU in which the primary PATHMON process is running. The location of the TCP backup process is not changed in this case.
3197 TCP <i>tcp-name</i> , TCP MEMORY DUMP NOT TAKEN - (10)	An internal error has occurred, but 10 dump files already exist. The TCP does not create more dump files in this situation.

Examples

The following command stops the TCP backup process for TCP-1 and restarts it in CPU 5:

```
CONTROL TCP TCP-1, BACKUPCPU 5
```

The following command sets the TCP DUMP attribute to OFF:

```
CONTROL TCP TCP-1, DUMP OFF
```

The following command changes the location of the TCP backup process and sets its DUMP attribute to ON:

```
CONTROL TCP TCP-1, BACKUPCPU 4, DUMP ON (FILE T2DUMP)
```

DELETE TCP Command

Use the DELETE TCP command to remove a TCP description from the PATHMON configuration file. A TCP must be in the STOPPED state before it can be deleted by the PATHMON process.

```
DELETE [ TCP ] { tcp-name
                  { ( tcp-name [ , tcp-name ] ... ) } }
```

tcp-name

specifies the name of a TCP. Use either a single name or several names separated by commas and enclosed in parentheses.

Examples

The following command deletes the TCP named TCP-1:

```
DELETE TCP TCP-1
```

The following command deletes three TCPs:

```
DELETE (TCP-3 , TCP-8 , TCP-9)
```

INFO TCP Command

Use the INFO TCP command to display the attribute values defined in the PATHMON configuration file for a TCP. This command does not display information about external TCPs (TCPs that are controlled by another PATHMON process and are therefore outside this PATHMON environment).

```
INFO [ / OUT list-file / ]

{ [ TCP ] tcp-name
  { [ TCP ] (tcp-name [ , tcp-name ]...) }
  { TCP * [ , option [ , option ] ] }

[ , OBEYFORM ]

option is:

    STATE state-type
    OBEYFORM
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

tcp-name

specifies the name of a TCP. You can use either a single TCP name or several TCP names separated by commas and enclosed in parentheses.

TCP *

displays the attribute values for all TCPs described in the PATHMON configuration file. If *option* is specified, displays the attribute values for all TCPs meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TCPs. Possible values are:

```
PENDING    NOT PENDING
RUNNING    NOT RUNNING
STOPPED    NOT STOPPED
```

This option displays the attribute values for TCPs in any state.

You can substitute SEL for STATE.

OBEYFORM

displays the information in the format you would use to set up a PATHMON configuration file; each attribute is displayed as a syntactically correct PATHCOM SET command. PATHCOM adds a RESET TCP command before and an ADD TCP command after each TCP description.

Considerations

- INFO TCP displays both the TCP attribute values you explicitly defined for a TCP and default values for the attributes you did not explicitly define. The attributes are displayed in alphabetic order. This command returns information about a TCP only after the description has been added to the PATHMON configuration file.
- Use the STATS TCP command to display the actual resources that are allocated after a TCP is running.
- OBEYFORM is an INFO command option for the the PATHMON process, PATHWAY, TCP, TERM, SERVER, and PROGRAM objects. By using the OUT *list-file* option and issuing an INFO command for each object in succession, you can build a new cold start command file that reflects any modifications you have made to your PATHMON configuration. You must add the START PATHWAY command at the end of the OUT file before using the file to start the PATHMON environment.
- The INFO TCP command displays the values specified for the SWAP and GUARDIAN-SWAP parameters, or the PATHMON-defined default values of these parameters if no values were specified. However, these values (specified or default) are not actually used, because the Kernel Managed Swap Facility (KMSF) now handles allocation of swap files for the TCP.

Examples

The following command requests information for two TCPs:

```
INFO (TCP-3 ,TCP-5)
```

The following command:

```
INFO TCP TCP-1 , OBEYFORM
```

causes the PATHMON process to display the TCP attributes in this format:

```
RESET TCP
SET TCP AUTORESTART 0
SET TCP CHECK-DIRECTORY ON
SET TCP CPUS 1:0
.      .
.      .
.      .
SET TCP PROGRAM \TS.$SYSTEM.SYSTEM.PATHTCP
SET TCP SERVERPOOL 20000
SET TCP STATS ON
```

```
SET TCP TCLPROG \*. $MFG.PARTS.POBJ
SET TCP TERMBUF 1500
SET TCP TERMPPOOL 10000
ADD TCP TCP-1
```

The following command directs the information to a particular file:

```
INFO/OUT TCPFILE/TCP *, OBEYFORM
```

The following command displays attribute values for all TCPs that are not in the PENDING state (that is, all TCPs in the RUNNING or STOPPED state) and uses the OBEYFORM display option:

```
INFO TCP *, STATE NOT PENDING, OBEYFORM
```

The following sequence of commands captures a configuration for a complete PATHMON environment (minus the START PATHWAY command) in the command file called NEWCONFIG:

```
=INFO/OUT NEWCONFIG/PATHMON, OBEYFORM
=INFO/OUT NEWCONFIG/PATHWAY, OBEYFORM
=INFO/OUT NEWCONFIG/TCP *, OBEYFORM
=INFO/OUT NEWCONFIG/TERM *, OBEYFORM
=INFO/OUT NEWCONFIG/SERVER *, OBEYFORM
=INFO/OUT NEWCONFIG/PROGRAM *, OBEYFORM
```

Note. For information about the PATHMON, PATHWAY, and SERVER objects listed in this example, see the *TS/MP System Management Manual*.

PRIMARY TCP Command

Use the PRIMARY TCP command to resume operation of the TCP primary process in the CPU defined for this process in the PATHMON configuration file. Use this command after an individual process or CPU failure, or after a SWITCH command has moved the TCP primary process from one CPU to another.

```
PRIMARY { [ TCP ] { tcp-name
              { (tcp-name [ , tcp-name ] ... ) } }
          { TCP *
          [ , IFPRICPU number ]
```

TCP *tcp-name*

specifies the name of a TCP running in your PATHMON environment.

IFPRICPU *number*

specifies the CPU in which one or more TCP primary processes are to run. If this CPU is defined for the TCP in the PATHMON configuration file, the PATHMON process resumes operation of the TCP primary process in this CPU.

If the TCP primary process is already running in this CPU, or if this is not the CPU configured for the TCP primary process, the PATHMON process ignores the command.

If you omit this option, the TCP primary process resumes operation in the CPU recorded in the PATHMON configuration file. If this CPU is down or the TCP backup process is not running, the PATHMON process returns an error.

TCP *

changes the location of all the TCP primary processes that are running in other than their configured primary CPUs. If *option* is specified, changes the location of all TCPs meeting the criteria specified.

Consideration

When the PRIMARY TCP operation completes, the PATHMON process returns the message:

```
INFO *2054* - TCP tcp-name, PRIMARIED
```

Errors

The following table lists the most common errors that can occur during the processing of the PRIMARY TCP command:

This Message...	Is Displayed When...
1070 BACKUP PROCESS NOT UP	You have issued a CONTROL TCP, BACKUPCPU command or a SWITCH TCP command directly after issuing a PRIMARY TCP command and the TCP backup process has not had enough time to be restarted.

Examples

The following command resumes operation of the TCPZ primary process in its configured CPU:

```
PRIMARY TCP TCPZ
```

The following command resumes operation of the TCPA primary process in CPU 2. However, operation is resumed only if 2 is the primary CPU value (CPUS 2:n) recorded for TCPA in the PATHMON configuration file.

```
PRIMARY TCPA, IFPRICPU 2
```

The following command changes the primary processes of all TCPs in the RUNNING state and designates CPU 2 only if CPU 2 is the primary CPU value (CPUS 2:n) recorded for all TCPs in the PATHMON configuration file:

```
PRIMARY TCP *, IFPRICPU 2
```

REFRESH-CODE TCP Command

Use the REFRESH-CODE TCP command to make the TCP read the SCREEN COBOL directory file to check for the latest version of a called program. This command is effective only if the TCP is running and the CHECK-DIRECTORY attribute is set to OFF. Use this command when a new version of a SCREEN COBOL program (in the TCLPROG file) becomes available to the TCP.

```
REFRESH-CODE { [ TCP ] tcp-name
                { [ TCP ] ( tcp-name [, tcp-name ]... )
                { TCP * [ , STATE state-type ] }
```

TCP *tcp-name*

specifies the name of a TCP running in your PATHMON environment.

TCP *

modifies the directory-checking operations of all TCPs running in this PATHMON environment. If STATE is specified, modifies all TCPs meeting the criteria specified.

STATE *state-type*

specifies the state of multiple TCPs. Possible values are:

PENDING	NOT PENDING
RUNNING	NOT RUNNING
STOPPED	NOT STOPPED

This command modifies directory-checking operations only for those TCPs in the RUNNING state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

Considerations

- In production systems, use the REFRESH-CODE TCP command to add a new version of a SCREEN COBOL program to the application.
- The REFRESH-CODE TCP command operates in conjunction with the TCP CHECK-DIRECTORY attribute. When the attribute is set to OFF, the TCP reads the TCLPROG directory file during a SCREEN COBOL CALL statement only if one of the following conditions exists:
 - The called program is not in the TCP code area.
 - The called program is in the TCP code area, but a REFRESH-CODE TCP command has been issued since the last time a call was made to the referenced program unit.

Note. The CHECK-DIRECTORY attribute affects the TCLPROG file at the TCP level only; it has no effect on any TCLPROG file at the TERMINAL or PROGRAM level. In other words, when this attribute is set to OFF, any TCLPROG file at the TERMINAL or PROGRAM level is still checked.

- If the TCP CHECK-DIRECTORY attribute is set to ON (ON is the default), the TCP always checks the TCLPROG directory file for the latest program version when executing a SCREEN COBOL CALL statement.

In a development environment where the SCREEN COBOL programs are often changed, having the TCP always check the TCLPROG directory file is good practice.

In a production system, it is recommended for performance reasons that you set CHECK-DIRECTORY to OFF and use the REFRESH-CODE TCP command when SCREEN COBOL programs have changed.

- If a TCP primary process fails and the CHECK-DIRECTORY attribute for that TCP is set to OFF, the TCP automatically reads the TCLPROG directory file during a SCREEN COBOL CALL statement before the TCP backup process takes over. This action occurs even if a REFRESH-CODE TCP command is not issued.

Examples

The following command modifies directory checking for the three named TCPs:

```
REFRESH-CODE TCP (TCP-1, TCP-4, TCP-6)
```

The following command modifies directory checking for all TCPs in the system:

```
REFRESH-CODE TCP *
```

The following command modifies directory checking for all TCPs in the RUNNING state:

```
REFRESH-CODE TCP *, STATE RUNNING
```

RESET TCP Command

Use the RESET TCP command to change the values for the TCP attributes (listed under *tcp-keyword* in the syntax) from the values you defined with the SET TCP command to the PATHMON defaults. This command does not change the attributes of a TCP already added to the PATHMON configuration file.

```
RESET TCP [ tcp-keyword [ , tcp-keyword ]... ]
```

tcp-keyword is:

AUTORESTART	INSPECT	POWERONRECOVERY
CHECK-DIRECTORY	MAXINPUTMSGLEN	PRI
CODEAREALEN	MAXINPUTMSG	PROCESS
CPUS	MAXPATHWAYS	PROGRAM
DEBUG	MAXREPLY	SENDMSGTIMEOUT
DUMP	MAXSERVERCLASSES	SERVERPOOL
GUARDIAN-LIB	MAXSERVERPROCESSES	STATS
GUARDIAN-SWAP	MAXTERMDATA	SWAP
HIGHPIN	MAXTERMS	TCLPROG
HOMETERM	NONSTOP	TERMBUF
		TERMPPOOL

tcp-keyword

specifies one or more attributes to be reset.

If you omit *tcp-keyword*, this command resets all attribute values to the PATHMON defaults.

Considerations

- If you invoke the RESET TCP command after the SET TCP command but before the ADD TCP command, the attribute values you specify revert to the PATHMON default values.
- Some required SET TCP attributes have no default values. If you include a required attribute in the RESET TCP command, the attribute is set to a null value and must be set again before you add the TCP description to the PATHMON configuration file.

If an attribute is not required and does not have a default value, the RESET TCP command deletes the value for the attribute.

Examples

The following command resets all attribute values:

```
RESET TCP
```

The following commands reset the values for the specified attributes:

```
RESET TCP MAXPATHWAYS , TERMBUF  
RESET TCP PRI , TERMPPOOL
```

SET TCP Command

Use the SET TCP command to establish the values for the TCP object attributes.

See [Section 2, Configuring Pathway/iTS Objects](#), for more information on configuring TCPs.

```
SET TCP tcp-attribute [ , tcp-attribute ]...
```

tcp-attribute is:

```

MAXTERMS number
TCLPROG file-name
AUTORESTART number
CHECK-DIRECTORY { ON | OFF }
CODEAREALEN double-number
CPUS primary : backup
DEBUG { ON | OFF }
DUMP { ON [ ( FILE file-name ) ] | OFF }
GUARDIAN-LIB file-name
GUARDIAN-SWAP $volume
HIGHPIN { ON | OFF }
HOMETERM file-name
INSPECT { ON [ ( FILE file-name ) ] | OFF }
LIKE tcp-name
MAXINPUTMSGLEN number
MAXINPUTMSGGS number
MAXPATHWAYS number
MAXREPLY bytes
MAXSERVERCLASSES number
MAXSERVERPROCESSES number
MAXTERMDATA bytes
NONSTOP { 0 | 1 }
POWERONRECOVERY { ON | OFF }
PRI priority
PROCESS $process-name
PROGRAM file-name
SENDMSGTIMEOUT { ON | OFF }
SERVERPOOL bytes
STATS { ON | OFF }
SWAP $volume
TERMBUF bytes
TERMPPOOL bytes
```

tcp-attribute

The following two attributes (the first two attributes listed above) have no defaults; you must specify them before adding a TCP definition:

MAXTERMS
TCLPROG

The rest of the attributes have default values.

see [Appendix C, Configuration Limits and Defaults](#), in this manual, and to the *TS/MP System Management Manual*, for information about configuration limits.

MAXTERMS *number*

specifies the maximum number of terminals that the TCP can have open at the same time. *number* must be a value from 0 through 4095, where 4095 is the total allowed for all PATHMON-controlled objects.

A TCP opens its terminals with a syncdepth of 1.

This attribute is required to create a valid TCP definition; there is no default value.

TCLPROG *file-name*

specifies the name of the SCREEN COBOL object library file that the TCP uses to locate the screen programs for its terminals. A file name consists of all characters prefixed to the letters COD in the name under which the SCREEN COBOL program is compiled. For example, if the object file name is \$VOL.SUBVOL.POBJCOD, *file-name* is POBJ.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the HP NonStop TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

If you specify a TCLPROG file name in either a SET TERM or SET PROGRAM command, the TCP first checks that file for the requested program unit before it checks the file specified in the SET TCP command.

This attribute is required to create a valid TCP definition; there is no default value.

AUTORESTART *number*

specifies the number of times that the PATHMON process attempts to restart the TCP within a fixed 10-minute interval after an abnormal termination, such as multiple CPU failures or the failure of an unpaired process.

number can be a value from 0 through 32,767. If you omit this attribute, the default is 0.

After an abnormal failure, the action caused by this option is equivalent to using a START TCP command followed by a START TERM command. This action affects all of the terminals configured and running with the AUTORESTART option at the time the TCP stops.

CHECK-DIRECTORY { ON | OFF }

specifies whether the TCP checks the SCREEN COBOL TCLPROG directory file (for example, POBJDIR) for the latest version of a called program when executing a SCREEN COBOL CALL statement.

If you omit this attribute, the default is ON.

Note. This attribute affects the TCLPROG file at the TCP level only; it has no effect on any TCLPROG file at the TERMINAL or PROGRAM level. In other words, when the CHECK-DIRECTORY attribute is set to OFF, any TCLPROG file at the TERMINAL or PROGRAM level is still checked.

ON

directs the TCP to check the SCREEN COBOL program directory file for the latest accessible version of the called program unit.

This is the recommended setting for a development environment in which SCREEN COBOL programs change frequently.

OFF

directs the TCP to check its memory for a version of the called program unit. If the program is already in the TCP code area and the TCP checked the directory file after the latest REFRESH-CODE command, the TCP uses that version of the program and does not read the directory file.

Note. This option does not ensure that the TCP uses the latest accessible version of the called program unit.

If the called program unit is not in memory, the TCP checks the directory file for the latest accessible version of the called program.

This is the recommended setting for a production environment. Pathway application performance is better when the TCP does not have to check the directory file every time a SCREEN COBOL program unit is called. Also, setting the CHECK-DIRECTORY attribute to OFF and using the REFRESH-CODE TCP command when a new version of a SCREEN COBOL program is put into production enables better tracking of SCREEN COBOL programs.

For more information about this option, see the [REFRESH-CODE TCP Command](#) on page 9-14.

CODEAREALEN *double-number*

specifies the number of bytes that the TCP allocates in its extended data segment for SCREEN COBOL object code.

double-number can be a value from 0 through 2,147,483,647. If you omit this value, the default is 65,536 bytes.

Note. The maximum value allowed for this parameter exceeds current system limitations, and the default value might be too small for some production environments. The code area length is limited by other aspects of your system, so that the practical maximum value is not a fixed number. Therefore, it is recommended that you estimate an appropriate value for the code area length according to the needs of your application.

The value you assign for this attribute depends on the number and the size of the screen programs in your applications and also on the available swap space. Some guidelines for estimating this attribute are:

- Use the size of the SCREEN COBOL programs (the screen data plus the RUNUNIT SIZE listed for these programs compiled with MAP) that make up the applications for this TCP. Note that not all of the screen programs are executed all of the time, and that the most regularly used programs determine the amount of code space most often required.
- Making the code area slightly larger than necessary can prevent the TCP from having to perform unnecessary code fetching during TCP operation. The larger code area only costs you disk space for the swap file.
- After running the TCP with STATS ON, check the STATS TCP display information and adjust the size of the code area to the actual value required. For more information about TCP STATS display information, see [Section 4, Maintaining Pathway/iTS Objects](#).

CPUS *primary:backup*

specifies the primary and backup processors on which the TCPs run.

You must set the TCP NONSTOP attribute to 1 to have a backup process created.

If you omit this attribute, the PATHMON process chooses a primary CPU and a backup CPU. This attribute is meaningless for single processor systems.

DEBUG { ON | OFF }

specifies whether a TCP enters debug mode on startup.

If you omit this attribute, the default is OFF.

ON

directs the TCP to enter debug mode. If the PATHMON process is configured with INSPECT, the TCP enters the HP *Inspect* symbolic debugger program instead of the DEBUG program.

OFF

directs the TCP not to enter debug mode.

DUMP { ON | OFF }

specifies whether the TCP writes the contents of its data stack and extended data segment information to a disk file if an internal or fatal TCP error occurs.

If you omit this attribute, the default is OFF

Note. The value defined for the CONTROL TCP DUMP attribute overrides the value defined in the SET TCP DUMP attribute.

ON

directs the TCP to create one or more disk files and write the contents of its data stack and its extended data segment to these files.

After creating 10 primary process and 10 backup process dump files, the TCP stops performing memory dumps. This prevents writing over existing dump files that may indicate the original cause of a problem, and prevents filling a disk with dump files.

FILE *file-name*

specifies the name of the file that the TCP creates for its dump operation. The format of the file name can be any of the following:

```
$volume.subvolume
$volume.subvolume.ZZ
$volume.subvolume.filename
```

For example, the following file name directs the TCP to create the appropriate ZZTCP_{nn}P or ZZTCP_{nn}B file on volume \$DATA using subvolume TCP05DMP:

```
(FILE $DATA.TCP05DMP)
(FILE $DATA.TCP05DMP.ZZ)
```

The following unique name for *file-name*, directs the primary TCP to name its dump file TCP05P and directs the backup TCP to name its dump file TCP05B:

```
(FILE $DATA.PWAYDMPS.TCP05)
```

Both the primary and backup dump files are created on volume \$DATA using subvolume PWAYDMPS. The P (primary TCP dump) and B (backup TCP dump) characters replace the last character of an eight-character name, or are appended to the end of a shorter dump file name. If the dump file already exists when the dump operation occurs, the TCPs reuse the existing file by purging the file's information and writing the latest memory dump.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the NonStop TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

If a file error interferes with creating a file that you specify, the TCP creates the dump file on the same subvolume as its program object file and uses the form ZZTCP_{nnx} for the file name.

If you do not specify *file-name*, the TCP creates its dump file on the same volume and subvolume as the PATHMON configuration file. The format of the file names are ZZTCP_{nn}P and ZZTCP_{nn}B, where *nn* is a number from 01 to 10, and P and B indicate the primary or the backup process, respectively, that the TCP dumped.

OFF

directs the TCP not to write data stack information to a file when it encounters an internal or fatal error.

GUARDIAN-LIB *file-name*

specifies the name of the library object file that contains additional TCP code space for user conversion routines. If you supply a value for this attribute, the operating system uses this file name and the file named in the PROGRAM attribute to access the TCP object code. For example:

```
SET TCP PROGRAM          $VOL1.SUBVOL1.PATHTCP2
SET TCP GUARDIAN-LIB     $VOL1.SUBVOL1.PATHTCPL
```

In this example, whoever starts the PATHMON process must have WRITE and EXECUTE access to the object file PATHTCP2.

After starting the TCP, the operating system modifies the TCP object file to point to the user library file that you define with this attribute. To change the user library file name, you must stop the TCP and alter this attribute value. Stopping the TCP does not change the user library pointer in the TCP object file.

All TCPs running simultaneously and using the same object code must also use the same user library object file. You can build the TCP user library object file by using the `nld` program development tool for combining object files. For more information, see the *HP NonStop Pathway/iTS TCP and Terminal Programming Guide*.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the *NonStop* TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

A PATHMON default value for this attribute does not exist. If you omit this attribute, the operating system uses the library file specified in the program object file PATHTCP2. Usually this file is on \$SYSTEM.SYSTEM.PATHTCPL.

The TCP is capable of supporting double-byte character devices, including 6530 devices and IBM 3270 devices. To enable double-byte character support in Pathway/iTS, you need to have product T9115, Multibyte Character Services Base (MBCS).

GUARDIAN-SWAP *\$volume*

specifies the *Guardian* operating environment name of the disk volume for the swap file formerly used by the operating system for memory swaps of the TCP data area (that is, user data not in the TCP extended data segment). Allocation of this swap file is now handled by the Kernel Managed Swap Facility (KMSF) on behalf of the TCP, and the value of this parameter is not used. However, if you specify a value for this parameter, it must represent an existing device, or an error occurs.

HIGHPIN { ON | OFF }

specifies whether the TCP runs at a high PIN or a low PIN.

ON The TCP runs at a high PIN.

OFF The TCP runs at a low PIN.

If you omit this attribute, the default is OFF.

For a TCP to run at a high PIN, the ?HIGHPIN flag must be set in the PATHTCP2 file. The ?HIGHPIN flag is set by default for this file.

HOMETERM *file-name*

specifies the name of the terminal that receives the TCP debugging information.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the NonStop TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

If you omit this attribute, the default is the home terminal of the PATHMON process

.

INSPECT {ON | OFF}

specifies whether you can use the Inspect program to examine the SCREEN COBOL programs running on the terminals controlled by a TCP. The SET TCP INSPECT attribute must be set to ON.

If you omit this attribute, the default is OFF.

ON

directs the TCP to allow communication between the Inspect program and the SCREEN COBOL programs after system startup.

FILE *file-name*

specifies the name of the Inspect command terminal.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the NonStop TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

If you omit *file-name*, the default is the home terminal of the TCP.

OFF

directs the TCP not to allow communication between the Inspect program and the SCREEN COBOL programs.

The TCP can control a maximum of 8 terminals simultaneously running the Inspect program and can process a total of 20 program breakpoints for all SCREEN COBOL programs under inspection.

If a TCP primary process fails, all breakpoints set for the Inspect program are lost. The breakpoints should be reentered when the TCP backup process takes over.

LIKE *tcp-name*

sets the attribute values for a TCP to those of the named TCP. *tcp-name* must be the name of a previously added TCP.

MAXINPUTMSGLEN *number*

specifies the maximum length in bytes of any unsolicited message that the TCP accepts. This does not include the header length.

number can be a value from 0 through 6000 bytes. If you omit this attribute, the default is 133 bytes. (Note that any bytes you specify in excess of the default are taken from the terminal pool area.)

The TCP rejects messages received that exceed MAXINPUTMSGLEN.

MAXINPUTMSGSGS *number*

specifies the maximum number of unsolicited messages that the TCP queues at any one time for all its requesters. When this attribute is exceeded, all arriving messages are rejected by the TCP until a terminal issues a REPLY to process messages already in the queue.

number must be a value from 0 to 2045. If you omit this attribute, the default is 0.

MAXPATHWAYS *number*

specifies the maximum number of external PATHMON processes (PATHMON processes controlling different PATHMON environments) that the TCP can communicate with at one time.

number must be a value from 0 through 4095. If you omit this attribute, the value of MAXPATHWAY is 0 and attempts by the TCP to access a server process in another PATHMON environment fail.

MAXREPLY *bytes*

specifies the maximum number of bytes permitted for an outgoing SEND message or a server reply message.

For a TCP that runs the gateway requester program to support Pathway/iTS web clients, set *bytes* to 32000. Otherwise, set *bytes* to the larger of either:

- The longest outgoing message (in bytes) from any SEND verb in any terminal under control of this TCP.

- The longest reply (in bytes) possible from any server process replying to a SEND verb from any terminal under control of this TCP.

If you omit this attribute, the default is 2000 bytes.

The TCP uses this number and the value for the MAXTERMDATA attribute to allocate the size of the context areas (Slot 0 and Slot 1) for its terminals.

MAXSERVERCLASSES *number*

specifies the maximum number of server classes with which a TCP can communicate simultaneously.

number must be a value from 0 through 4095, where 4095 is the total allowed for all PATHMON-controlled objects. If you omit this attribute, the default is the number of server classes specified in the SET PATHWAY command.

Note. For details about the SET PATHWAY command, see the *TS/MP System Management Manual*.

MAXSERVERPROCESSES *number*

specifies the maximum number of links a TCP can have to all servers in all server classes.

number must be a value from 0 through 4095, where 4095 is the total allowed for all PATHMON-controlled objects. If you omit this attribute, the default is the number of server processes specified in the SET PATHWAY command.

MAXTERMDATA *bytes*

specifies the number of bytes that the TCP allocates for context data for each terminal. The TCP uses this value and the value of MAXREPLY to determine the size of a terminal's slot areas. (Slot 0 and Slot 1 are the same size.)

For a TCP that runs the gateway requester program to support Pathway/iTS web clients, set *bytes* to 200000.

The maximum value is 2,147,483,647 bytes. The minimum value is 2804 bytes if the NONSTOP attribute is set to 1. If you enter a value that is less than 2804 bytes and the NONSTOP attribute is set to 1, the PATHMON process automatically increases the value to the minimum value. There is no minimum value if the NONSTOP attribute is set to 0.

If you omit this attribute, the default is 8000 bytes.

After running the TCP, check the STATS TERM display (the AREA INFO: MAX SIZE, DATA value) to find the actual size of the context data area that the TCP allocates for a terminal. For more information on determining the actual size of the context data area, see [Section 5, Tuning Your System Using Statistics](#).

NONSTOP { 0 | 1 }

specifies whether a TCP runs with a backup process and performs normal checkpoint operations.

0 The TCP does not run with a backup process.

1 The TCP runs with a backup process.

If you omit this attribute, the default is 1. In a single processor environment, using the default generates an error. You can either ignore the error or set NONSTOP to 0.

POWERONRECOVERY { ON | OFF }

specifies whether or not a TCP automatically recovers a screen after it encounters a file-system error 191.

This attribute is only valid for 6530 devices in block mode. A 3270 device causes a modem to disconnect when power is turned off.

If you omit this attribute, the default is ON.

ON

directs the TCP to recover the screen automatically—except for data and display characteristics—by first executing a DISPLAY BASE statement for the current base screen and a DISPLAY OVERLAY statement for every current overlay screen.

After all the current screens are displayed, the TCP checks for a user exception code declared in the USE FOR SCREEN RECOVERY clause. If the user exception code is available, the TCP transfers control to it. If the code is not available, the TCP returns control to the ACCEPT statement.

OFF

directs the TCP not to recover the screen automatically. Instead, the TCP displays a blank screen ready to read a function key entry.

PRI *priority*

specifies the priority at which a TCP runs.

priority can be a value from 1 through 199. If you omit this attribute, the default is 20 less than the priority of the PATHMON process.

PROCESS *\$process-name*

specifies the process name of the TCP when it starts.

process-name can be a maximum of 5 characters, beginning with a dollar sign (\$) and followed by 1 to 4 alphanumeric characters. The first character must be a letter.

If you omit this attribute, the default is in the format \$Z_{nnn}.

PROGRAM *file-name*

specifies the TCP object file name (usually \$SYSTEM.SYSTEM.PATHTCP2).

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the NonStop TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

If you omit this attribute, the default is \$SYSTEM.SYSTEM.PATHTCP2.

SENDMSGTIMEOUT { ON | OFF }

specifies whether or not the TCP should filter out 3161 timeout error messages when the ON ERROR clause is used with a SEND MESSAGE statement.

ON The TCP does **not** filter out 3161 timeout messages.

OFF The TCP filters out 3161 timeout messages when the ON ERROR clause is used with a SEND MESSAGE statement.

When the ON ERROR clause is used with a SEND MESSAGE statement, only the first 3161 timeout encountered by each TCP configured under a PATHMON is logged. An additional 3161 timeout is logged when the primary TCP fails and the backup TCP takes over and encounters the timeout. The TCPs filter out all other 3161 timeout errors. If the ON ERROR clause is not used and a 3161 timeout occurs, the corresponding terminal is suspended after the timeout is logged.

If you omit this attribute, the default is ON.

SERVERPOOL *bytes*

specifies the number of bytes that the TCP allocates for I/O requests and replies between the SCREEN COBOL programs and server processes. Space is also allocated for receiving and sending unsolicited messages.

The value for *bytes* is the maximum number of bytes that the TCP allocates for a server I/O buffer and must be at least the size of the largest predefined server reply message.

For a TCP that runs the gateway requester program to support Pathway/iTS web clients, set the value of this attribute to 40,000 bytes. Otherwise, the suggested range is 10,000 to 30,000 bytes. The minimum value is 32 bytes. If you omit this attribute, the default is 20,000 bytes.

The maximum useful value for SERVERPOOL is the length of the largest request or reply multiplied by the number of terminals that the TCP controls. If the program uses unsolicited message processing (UMP), add in the value of the maximum concurrent UMP I/O operation multiplied by the maximum UMP message size.

See [Section 5, Tuning Your System Using Statistics](#), for more details about the SERVERPOOL attribute.

Note. The pool space value allocated for SERVERPOOL is the value you configure rounded up to the next highest multiple of 4.

STATS { ON | OFF }

specifies whether the TCP gathers resource usage and system performance statistics.

ON The TCP gathers statistics.

OFF The TCP does not gather statistics.

If you omit this attribute, the default is OFF.

This attribute works in conjunction with the STATS command, which displays TCP, TERM, and SERVER object statistics gathered by the TCP.

SWAP *\$volume*

specifies the disk volume name for the temporary file formerly created and managed by the operating system for memory swaps of the TCP extended data segment. Allocation of this swap file is now handled by the Kernel Managed Swap Facility (KMSF) on behalf of the TCP, and the value of this parameter is not used. However, if you specify a value for this parameter, it must represent an existing device, or an error occurs.

TERMBUF *bytes*

specifies the maximum number of bytes that the TCP allocates from the TERMPOOL area for its terminal output buffers. (This allocation has no effect on terminal input operations.)

The value for *bytes* is a value from 256 to the value of the TERMPOOL attribute. If you enter a value for TERMBUF that is less than 256, the PATHMON process automatically adjusts the value to 256. If you omit this attribute, the default is 1500 bytes.

For a TCP that runs the gateway requester program to support Pathway/iTS web clients, set the value of this attribute to 32,000 bytes.

If you are writing diagnostic display messages to the terminal, *bytes* should be a value from 1315 to the value of the TERMPOOL attribute. Diagnostic display messages are not written to the terminal if the value is less than 1315.

Defining a large value for this attribute might decrease the number of terminals able to operate simultaneously or increase the number of times an I/O operation is delayed until space for a buffer becomes available. Small values tend to increase the number of output operations necessary to perform certain functions (for example, the execution of the SCREEN COBOL DISPLAY BASE statement).

These buffers, like any pool buffer, are held only for the time required to perform the terminal I/O operation.

Note. TERMBUF is not relevant to verbs (for example, the SEND MESSAGE verb) used with the IDS facility.

TERMPOOL *bytes*

specifies the number of bytes that the TCP allocates in its data area for all terminal I/O buffers.

The value for *bytes* is the minimum number of bytes that the TCP allocates for the pool size. If you omit this attribute, the default is 10,000.

For a TCP that runs the gateway requester program to support Pathway/iTS web clients, set *bytes* to 200,000. Otherwise, calculate an appropriate TERMPOOL value for your configuration by multiplying the number of TERM objects by the size of the largest terminal input buffer plus the largest UMP message. For example, with 100 TERM objects, a largest terminal input buffer of 1000 bytes, and a largest UMP message of 1000 bytes, set the TERMPOOL value to 200,000 bytes ($100 \times (1000 + 1000) = 200,000$).

If you specify a smaller value for TERMPOOL than the above calculation would provide, make sure to specify a SET TCP TERMPOOL value of at least 25 bytes greater than the SET TCP TERMBUF value; the TCP requires the additional 25 bytes for pool management.

Note. The pool space value allocated for TERMPOOL is the value you configure rounded up to the next highest multiple of 4.

Considerations

- Before adding a TCP to the PATHMON configuration file, you must define values for all of the required attributes listed in this command.
- If you repeat a SET TCP command with a different attribute value, PATHCOM uses the last value entered for the TCP attribute.
- If you specify the user library object file name, then you should make the GUARDIAN-LIB attribute a standard part of your PATHMON environment startup procedure; this ensures that the proper object file is always used. If you customize your user library object file or if you install the file in a location other than \$SYSTEM.SYSTEM.PATHTCPL, then you must remember to update your library whenever HP adds new procedures to the standard user library. Updating your library is essential because new versions of the TCP may invoke the new procedures in the user library.

Errors

The following table lists the most common errors that can occur during the processing of the SET TCP command:

This Message...

1038 TCP *tcp-name*, PROCESS
CREATION FAILURE: FILE SYSTEM
ERROR ON LIBRARY FILE -
SECURITY VIOLATION (48)

1038 TCP *tcp-name*, PROCESS
CREATION FAILURE : LIBRARY
CONFLICT

3239 - TCP *tcp-name*, GUARDIAN-
LIB INCOMPATIBLE WITH TCP

Is Displayed When...

The person starting the PATHMON process
does not have WRITE access to the file specified
in the PROGRAM or GUARDIAN-LIB attributes.

The associated TCP is not started.

You tried to start a TCP having a GUARDIAN-
LIB file name different from the file name being
used by the currently running TCPs.

The TCP tried to use a new procedure in the
standard user library, but the new procedure was
not present.

Use the GUARDIAN-LIB attribute to update your
user library.

Example

The following commands define a subset of the TCP attributes:

```
SET TCP CPUS 1:2
SET TCP GUARDIAN-LIB $SYSTEM.SYSTEM.PATHTCPL
SET TCP GUARDIAN-SWAP $SYTM1
SET TCP INSPECT ON (FILE $CMDT)
SET TCP MAXPATHWAYS 4
SET TCP MAXTERMS 15
SET TCP PROGRAM $SYSTEM.SYSTEM.PATHTCP2
SET TCP TCLPROG $MFG.INVEN.POBJ4
```

SHOW TCP Command

Use the SHOW TCP command to display a subset of the TCP attributes in alphabetic order. This command displays the attribute values for a TCP not yet added to the PATHMON configuration file.

```
SHOW [ / OUT list-file / ] TCP
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

Considerations

- When PATHCOM first starts or after a RESET TCP command executes, the SHOW TCP command displays only the required SET TCP attributes and a subset of the optional attributes that have PATHMON defaults. A question mark appears for the required attribute values that you must set before adding the TCP.
- SHOW TCP displays all of the attribute values that you explicitly set with the SET TCP command.
- For attributes that include Guardian file names, SHOW TCP displays the node-name part of the file name as either *\nodename* or ***. The *\nodename* form is used if the node was explicitly specified by the user. The *** form is used if the attribute was specified to default to the node where the PATHMON process is running.
- After an ADD TCP command, SHOW TCP continues to display the attribute values held in PATHCOM for the previously added TCP. Use the RESET TCP command to change all of the values to their defaults, or use the SET TCP command to replace specific attribute values.
- After a PATHCOM OPEN command, the SHOW TCP command displays the TCP attribute values PATHCOM holds for the PATHMON environment controlled by the PATHMON process you have opened. Use the SET TCP command to assign the attribute values you require.

Examples

If you issue the SHOW TCP command before setting the TCP attributes, the following information is displayed:

```
TCP
  AUTORESTART 0
  CHECK-DIRECTORY ON
  DEBUG OFF
  .
  .
  .
  TCLPROG ?
  TERMBUF 1500
  TERMPPOOL 10000
```

The following command directs the SHOW command output to a file named TCPFLE:

```
SHOW/OUT TCPFLE/TCP
```

START TCP Command

Use the START TCP command to initiate operation of TCPs. This command cannot initiate an external TCP (a TCP that is controlled by another PATHMON process and is therefore outside this PATHMON environment). If the primary processor defined for a TCP is unavailable, the PATHMON process starts the TCP in the backup processor defined for that TCP.

```
START { [ TCP ] tcp-name
        { [ TCP ] ( tcp-name [ , tcp-name ] ... )
        { TCP * [ , STATE state-type ] }
```

TCP *tcp-name*

specifies the name of one or more previously defined and added TCPs. *tcp-name* can be either a single TCP name or several TCP names separated by commas and enclosed in parentheses.

TCP *

starts all TCPs in the PATHMON configuration file.

STATE *state-type*

specifies the state of the multiple TCPs. Possible values are:

```
PENDING    NOT PENDING
RUNNING    NOT RUNNING
STOPPED     NOT STOPPED
```

This command starts only those TCPs in the STOPPED state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

Errors

The following table lists the most common error that can occur during the processing of the START TCP command:

This Message...	Is Displayed When...
FILE OPEN ERROR - <i>pathmon-name</i> - FILE IN USE (12)	<p>You have issued a START TCP command but too many processes are running concurrently under the PATHMON process and the PATHMON process cannot start the TCP.</p> <p>There is a limit to the number of concurrently running processes (TCPs, external TCPs, LINKMONs, PATHCOMs, and SPI processes) that a PATHMON process can handle. See Appendix C, Configuration Limits and Defaults, for more information on configuration limits and defaults.</p> <p>Reissue the START TCP command; once one of the other processes stops, the PATHMON process can start the TCP.</p>

Examples

The following commands start only the TCPs specified:

```
START TCP TCP-0
START (TCP, TCP3, TCP4)
```

The following command starts all of the TCPs defined in the PATHMON configuration file:

```
START TCP *
```

The following command starts all TCPs in the STOPPED state:

```
START TCP *, STATE STOPPED
```

STATS TCP Command

Use the STATS TCP command to display resource usage and performance statistics collected by the TCP. These statistics provide information about TCP operations and data space allocations. This command does not display statistics for any external TCPs (TCPs that are controlled by another PATHMON process and are therefore outside this PATHMON environment) that might be linked to local server processes. The STATS attributes of the SET TCP command and the CONTROL TCP command control the gathering of these statistics.

```
STATS [ / OUT list-file / ]

{ [ TCP ] tcp-name }
{ [ TCP ] ( tcp-name [ , tcp-name ]... ) }
{ TCP * [ , option [ , option ]... ] }

[ , tcp-attribute [ , tcp-attribute ]... ]
```

option is:

```
STATE state-type
COUNT number
DETAIL
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET
```

tcp-attribute is:

```
COUNT number
DETAIL
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

tcp-name

specifies one or more TCPs. *tcp-name* can be either the name of a single TCP or several TCP names separated by commas and enclosed in parentheses.

TCP *

displays statistics for all TCPs described in the PATHMON configuration file. If *option* is specified, displays statistics for all TCPs meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TCPs. Possible values are:

PENDING	NOT	PENDING
RUNNING	NOT	RUNNING
STOPPED	NOT	STOPPED

This command displays statistics only for those TCPs in the RUNNING state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

COUNT *number*

specifies the number of times that the STATS command repeats. You can stop the command from repeating by pressing the Break key.

DETAIL

includes statistics such as response time information, for the terminals and server classes associated with the specified TCP.

FREQTABLE

generates a frequency distribution table containing statistics for each terminal and server class associated with the specified TCP.

The frequency distribution table is not generated—even if you specify the FREQTABLE option—under these conditions:

- There have been fewer than 50 response time measurements collected at the time of the STATS request.
- At the fiftieth measurement, the size of the time interval calculated is less than 0.01 second.

For more information about the FREQTABLE option, including a sample display, see [Section 5, Tuning Your System Using Statistics](#).

INTERVAL *number*

specifies the time period between the statistics displays, incremented in hours, minutes, or seconds.

RESET

sets the counters used for the measurements back to zero.

Considerations

- If the SET TCP STATS attribute value is set to OFF, the TCP does not gather information for this command.
- When you use the DETAIL option, the TCP displays the terminals and server class statistics described in [Section 10, TERM Commands](#), and in the *TS/MP System Management Manual*, respectively.
- For more information about PATHMON environment statistics and the STATS TCP command display, see [Section 5, Tuning Your System Using Statistics](#), and the *TS/MP System Management Manual*.

Examples

The following command displays the statistical information gathered for the three named TCPs for a 10-minute interval:

```
STATS TCP-1, TCP-3, TCP-5, DETAIL, INTERVAL 10 MINS
```

The following commands direct statistical information, including measurements on system responsiveness (as perceived by the user), to a specified file:

```
STATS /OUT STATFLE/ TCP-1, DETAIL, FREQTABLE, INTERVAL 2 HRS
STATS /OUT STATFLE/ TCP *, RESET, INTERVAL 30 MINS, COUNT 2
```

The following command displays statistical information for all TCPs in the RUNNING state for a 10-minute interval:

```
STATS TCP *, STATE RUNNING, INTERVAL 10 MINS
```

STATUS TCP Command

Use the STATUS TCP command to display the status of a TCP and the terminals it controls.

```
STATUS [ / OUT list-file / ]

{ [ TCP ] tcp-name [ , DETAIL ] }
{ [ TCP ] ( tcp-name [ , tcp-name ]... ) [ , DETAIL ] }
{ [ TCP ] E\node.$process-name [ , DETAIL ] }
{ TCP * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
DETAIL
```

OUT *list-file*

directs output to the file that you specify; this could be a DEFINE name. If you omit this option, the PATHMON process writes the output to the PATHCOM list file; this is typically the home terminal.

tcp-name

specifies the name of a previously defined and added TCP.

E*node*.*\$process-name*

specifies the name of an external TCP (a TCP that is controlled by another PATHMON process and, therefore, is outside this PATHMON environment).

E indicates an external TCP name.

node specifies the name of the NonStop system on which the TCP is running. This can be the same or different from the system on which the PATHMON process is running.

\$process-name specifies the external TCP name.

DETAIL

includes status information on the terminals controlled by the specified TCPs. The DETAIL option does not display status information for external TCPs.

TCP *

displays the status of all local and external TCPs defined for the PATHMON environment.

STATE *state-type*

specifies the state of the multiple TCPs. Possible values are:

PENDING	NOT	PENDING
RUNNING	NOT	RUNNING
STOPPED	NOT	STOPPED

This option displays status information for TCPs in any state.

You can substitute SEL for STATE.

Display Format Without DETAIL

The format of the display returned by the STATUS TCP command without the DETAIL option is shown below.

TCP	STATE	ERROR	INFO	PROCESS	CPUS
<i>tcp-name</i>	<i>state</i>	<i>pw-error</i>	<i>info</i>	<i>process-name</i>	<i>pri:backup</i>
.
.
.

The fields in this display are as follows:

Display Field	Description
TCP	Name of the TCP
STATE	Current state of the TCP. Possible values are: STOPPED TCP is stopped PENDING TCP received a STOP command RUNNING TCP is running and capable of controlling terminals
ERROR	PATHMON environment error number
INFO	Additional information regarding the error number. See the error messages listed in Section 13, TCP Messages (Numbers 3000-3999) .
PROCESS	Guardian process name of the TCP
CPUS	Current primary and backup CPUs (which is not necessarily the same as those configured).

Display Format With DETAIL

The format of the display returned by the STATUS TCP command with the DETAIL option is shown below.

TCP	STATE	ERROR	INFO	PROCESS	CPUS
<i>tcp-name</i>	<i>state</i>	<i>pw-error</i>	<i>info</i>	<i>process-name</i>	<i>pri:backup</i>
.
.
.
TERM	WAIT	PENDING	ACCEPT	STOPMODE	TRANSMODE
<i>term-name</i>	<i>wait-cause</i>	<i>pend-cause</i>	<i>accept</i>	<i>stopmode</i>	<i>transmode</i>
				[RECOVERING]	
.
.
.

The TCP, STATE, ERROR, INFO, PROCESS, and CPUS fields have the same meanings as for the display without the DETAIL option. The additional fields returned in the detailed display are as follows:

Display Field	Description
TERM	Name of the terminal
WAIT	Reason the terminal is waiting. The possible values are listed separately below.
PENDING	Command that is pending to the terminal. The possible values are: <ul style="list-style-type: none"> SUSPEND A SUSPEND command has been issued to the terminal, but either the terminal's SCREEN COBOL special register STOP-MODE is nonzero or the terminal execution has not reached a state where it can be suspended. SUSPEND! A SUSPEND! command has been issued to the terminal, but terminal execution has not reached a state where it can be suspended. STOP A STOP command has been issued to the terminal, but either the terminal's SCREEN COBOL special register STOP-MODE is nonzero or the terminal execution has not reached a state where it can be suspended.
ACCEPT	Possible values are: <ul style="list-style-type: none"> YES Terminal execution is at an ACCEPT statement. YES-RESTART The terminal is in transaction mode; the transaction has been restarted.
STOPMODE	Current value of the terminal's SCREEN COBOL special register. Possible value is: <ul style="list-style-type: none"> NON-ZERO The value of the register is nonzero.
TRANSMODE	Whether or not the terminal is in transaction mode. Possible value is: <ul style="list-style-type: none"> YES The terminal is in transaction mode and the data is protected by TMF auditing. If the terminal is not in transaction mode, this field is left blank.
RECOVERING	Whether or not screen recovery is currently taking place for the terminal

The possible values for the WAIT field are as follows:

<i>wait-cause</i>	Description
READY	The terminal is ready.
ERMBUF	The terminal is waiting for internal terminal I/O buffer space (TERMPOOL) to become available.
SERVBUF	The terminal is waiting for internal server I/O buffer space (SERVERPOOL) to become available.
TERMWRITE	The terminal is waiting for a file-system write operation to the terminal to complete.

<i>wait-cause</i>	Description
TERMREAD	The terminal is waiting for a file-system read operation to the terminal to complete.
SERVER	The terminal is waiting for a server process to become available (all links to the desired server class are currently in use), or there are outstanding link requests to the PATHMON process.
SERVERIO	The terminal is waiting for a request to a server to complete.
MEMMAN	The terminal is waiting for an internal memory management request to complete.
ITMACQ	(Inter-Task Message ACQUIRE) The terminal is waiting to report an error to the PATHMON process or to obtain a tell message from the PATHMON process.
ITMCOMP	(Inter-Task Message COMPLETE) The terminal is waiting for the completion of an error report to the PATHMON process or the requested tell message from the PATHMON process.
TIMEOUT	The terminal is waiting for a period of time to elapse. Terminal waiting can happen during retry processing after a terminal I/O error or periodically during execution of a program that is performing internal processing when no I/O is occurring.
SUSP-RESTART	The terminal is suspended because of a nonfatal error or because a SUSPEND command was issued to the terminal. Terminal execution can be restarted by issuing a RESUME command.
SUSP-ABORT	The terminal is suspended because of a fatal error. Terminal execution cannot be restarted until an ABORT command is issued.
FROZEN	The terminal is waiting because it is currently attempting to communicate with a frozen server class.
BKPT-HOLD	The terminal task is at an Inspect breakpoint.
CHECKPOINT	The terminal is waiting for completion of a checkpoint.
END-TRANS	The terminal is waiting for the TMF subsystem to complete an END-TRANSACTION operation.
INSPECT-FILE	The terminal task is waiting to communicate with the Inspect process to perform some Inspect operation.
IMON-I/O	The terminal task is doing an I/O operation to the IMON process.
INSPECT-I/O	The terminal task is doing an I/O operation to the Inspect process.
PMCB-LOCK	(PATHMON Control Block LOCK) The TCP is waiting for the PATHMON process to become unlocked.
PATHMON-OPEN	The TCP is opening an external PATHMON process.
PATHMON-I/O	The TCP is waiting for completion of an I/O request to an external PATHMON process.
TERM-OPEN	The terminal task is doing an OPEN operation to a terminal or a terminal-simulation process.

Consideration

When a TCP from one PATHMON environment requests a link to a server class in a different PATHMON environment, the PATHMON process controlling the server class considers the requesting TCP to be an external process. However, a STATUS TCP command issued to the local PATHMON process (the one controlling the server class) displays the names of both the external and local TCPs. The external TCP name is generated by the PATHMON process at the time of the status request.

Example

The following example commands request status information for various TCPs using different options:

```
STATUS TCP-1
STATUS/OUT STATFLE/TCP (TCP-2,TCP-3)
STATUS TCP TCP-4,DETAIL
STATUS TCP *, STATE RUNNING
STATUS/OUT STATFLE/TCP *
STATUS TCP E\CUPRTNO.$TCP1
STATUS TCP *, STATE NOT PENDING
```

STOP TCP Command

Use the STOP TCP command to stop a TCP. This command does not stop an external TCP (a TCP that is controlled by another PATHMON process and is therefore outside this PATHMON environment) that is linked to a local server process.

```
STOP { [ TCP ] tcp-name [ , WAIT ] }
      { [ TCP ] ( tcp-name [ , tcp-name ]... ) [ , WAIT ] }
      { TCP * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
WAIT
```

tcp-name

specifies the name of one or more previously defined and added TCPs. *tcp-name* can be either a single TCP name or several TCP names separated by commas and enclosed in parentheses.

WAIT

directs the PATHMON process to retry the command every second if the STOP command is rejected with an invalid TCP-state error. You can abort the WAIT option by pressing the Break key.

If you omit this option, the STOP command aborts on any error.

TCP *

stops all TCPs configured in the PATHMON configuration file.

STATE *state-type*

specifies the state of the multiple TCPs. Possible values are:

PENDING	NOT PENDING
RUNNING	NOT RUNNING
STOPPED	NOT STOPPED

This command stops only those TCPs in the RUNNING state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

Considerations

- The PATHMON process stops all TCPs in the order that you specify. It cannot stop a TCP until all of the terminals controlled by the TCP are stopped.
- If you use the STOP WAIT option, the PATHMON process immediately stops all the TCPs that it can and stops any remaining TCPs later.

Examples

The following command directs the PATHMON process to stop all TCPs in the RUNNING state:

```
STOP TCP *, STATE RUNNING
```

The following command stops only the specified TCPs:

```
STOP TCP-1  
STOP TCP (TCP-3, TCP-4)  
STOP (TCP5, TCP6), WAIT
```

The following command stops all TCPs:

```
STOP TCP *, WAIT
```

SWITCH TCP Command

Use the SWITCH command to make the TCP exchange the function of its primary process with the function of its backup process while the TCP is running. This command does not change the value of the SET TCP CPUS attribute in the PATHMON configuration file.

```
SWITCH { [ TCP ] tcp-name
        { [ TCP ] tcp-name [ , tcp-name ] ... }
        { TCP * [ , STATE state-type ] }
```

TCP *tcp-name*

specifies the name of one or more previously defined and added TCPs. *tcp-name* can be either a single TCP name or several TCP names separated by commas and enclosed in parentheses.

The TCP must have an initialized backup process.

If the named TCP is not running or is not defined with the NONSTOP attribute on, an error is returned.

TCP *

exchanges the operations of all the primary and backup processes for TCPs defined with the NONSTOP attribute on. This option changes only those TCPs with initialized backup processes.

STATE *state-type*

specifies the state of the multiple TCPs. Possible values are:

```
PENDING    NOT PENDING
RUNNING    NOT RUNNING
STOPPED     NOT STOPPED
```

This command exchanges the operations of primary and backup processes only for those TCPs in the RUNNING state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

Considerations

- After this command completes, PATHCOM returns these information message:

```
INFO - *2031* TCP tcp-name SWITCHED
```

- If you restart a TCP that was running with its primary and backup processes switched, the primary and backup processes start in their configured CPUs.

Errors

The following table lists the most common errors that can occur during the processing of the PRIMARY TCP command:

This Message...	Is Displayed When...
1070 BACKUP PROCESS NOT UP	You have issued a CONTROL TCP, BACKUPCPU command or a SWITCH TCP command directly after issuing a PRIMARY TCP command and the TCP backup process has not had enough time to be restarted.

Examples

The following command directs the primary process for TCPA to operate as the backup process and the backup process to assume the operations of the primary process:

```
SWITCH TCP TCPA
```

The following command causes all TCPs that are in the RUNNING state to exchange primary and backup operations:

```
SWITCH TCP *, STATE RUNNING
```

Wild Card Support for TCP Commands

The following TCP commands in PATHCOM allow management and operation of all TCP objects configured under a PATHMON environment by using the wild card "*":

- INFO
- START
- STATS
- STATUS
- STOP
- SWITCH

All these commands can be applied to a subset of configured TCP objects.

Note. The above mentioned wild card support is applicable only to TS/MP 2.3 PATHMON and PATHCOM. For more information on TS/MP 2.3, see the *TS/MP Release Supplement Manual*.

Example

The following command displays the status information for all terminal control processes configured under the PATHMON environment whose name starts with TCP:

```
STATUS TCP TCP*
```

The system displays this output:

TCP	STATE	ERROR INFO	PROCESS	CPUS
TCP1	RUNNING		\ACS.\$Y75V	2:3
TCP2	RUNNING		\ACS.\$Y75V	2:3
TCPA	RUNNING		\ACS.\$Y75V	2:3
TCPB	RUNNING		\ACS.\$Y75V	2:3

10 TERM Commands

This section describes the PATHCOM commands that define and control configured TERM objects and control and inquire about temporary TERM objects; the commands are listed in alphabetical order. The commands are as follows:

To Perform the Following Tasks...	Use These PATHCOM Commands...
Define and remove TERMS	SET TERM ADD TERM DELETE TERM
Control TERMS	START TERM STOP TERM ABORT TERM SUSPEND TERM RESUME TERM CONTROL TERM
Change TERM attributes	ALTER TERM RESET TERM
Obtain information about	INFO TERM SHOW TERM STATS TERM STATUS TERM
Perform other management tasks	INSPECT TERM

These commands configure and control SCREEN COBOL programs that control input-output (I/O) devices, such as terminals and workstations, or control I/O processes that enable users to communicate with a transaction processing application. You must define a TERM object for each device. With these commands you can do the following:

- Set and modify TERM object attributes
- Add and delete TERM objects
- Start and stop TERM objects
- Display TERM object attribute and status information

See [Section 12, Tell Message Commands](#), for a description of the TELL TERM command.

ABORT TERM Command

Use the ABORT TERM command to abort a suspended or running TERM object.

This command can be issued to configured and temporary TERM objects.

```
ABORT { [ TERM ] term-name
        { [ TERM ] ( term-name [ , term-name ]... )
        { TERM * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
TCP tcp-name
```

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

TERM *

aborts all TERM objects in the PATHMON environment. If *option* is specified, stops all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

```
RUNNING      NOT RUNNING
STOPPED      NOT STOPPED
SUSPENDED    NOT SUSPENDED
```

This command aborts only those TERM objects in the RUNNING or SUSPENDED state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

TCP *tcp-name*

aborts all TERM objects controlled by the specified TCP.

Considerations

- If a TERM object is in TMF transaction mode when this command is executed, the transaction is backed out before the TERM aborts.
- This command is usually issued only to configured TERM objects; temporary TERM objects created in response to a RUN PROGRAM command are usually terminated automatically by the PATHMON process when the task has completed.

- The coupling information is lost after issuing an ABORT command.

Examples

The following commands abort the specified TERM objects:

```
ABORT TERM TERM-1
ABORT TERM-2
ABORT TERM (TERM3 , TERM4 )
ABORT (TERM-5 , TERM-6 )
```

The following command aborts all TERM objects in the PATHMON environment:

```
ABORT TERM *
```

The following command stops all TERM objects in the SUSPENDED state under the control of the TCP named TCPA:

```
ABORT TERM *, STATE SUSPENDED, TCP TCPA
```

ADD TERM Command

Use the ADD TERM command to enter the name and description of a TERM object into the PATHMON configuration file. Use this command after defining a TERM object with the SET TERM command.

This command can be issued only to configured TERM objects.

```
ADD TERM term-name [ , term-attribute ]...
```

term-name

specifies the name of a TERM object. A TERM name can be from 1 through 15 alphanumeric or hyphen characters and must start with a letter, be unique within the PATHMON environment, and not be a PATHCOM reserved word.

term-attribute

specifies an attribute describing a TERM object. A TERM attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET TERM command. Use any of the attributes listed for the SET TERM command.

Considerations

- The ADD TERM is valid only for configured TERM objects. Temporary TERM objects are added when the RUN PROGRAM command executes.
- To use the ADD TERM command, you must have a licensed copy of the HP NonStop Pathway/iTS product on your system; otherwise, the PATHMON process returns an error.

Examples

The following commands add definitions for the specified TERM objects to the PATHMON configuration file:

```
ADD TERM TERM-1  
ADD TERM TERM-4 ,LIKE TERM-1
```

ALTER TERM Command

Use the ALTER TERM command to change the attribute values of a TERM object that was previously added to the PATHMON configuration file. For a list of the TERM attributes, see the [SET TERM Command](#) on page 10-18.

The TERM object must be stopped before you can alter an attribute value.

This command can be issued only to configured TERM objects.

```
ALTER [ TERM ] term-name
```

```
{ , term-attribute [ , term-attribute ]... }
{ , RESET term-keyword }
{ , RESET ( term-keyword [ , term-keyword ]... ) }
```

term-keyword is:

AUTORESTART	FILE	TCLPROG
BREAK	INITIAL	TCP
DIAGNOSTIC	INSPECT	TMF
DISPLAY-PAGES	IOPROTOCOL	TRAILINGBLANKS
ECHO	MAXINPUTMSGs	TYPE
EXCLUSIVE	PRINTER	

term-name

specifies the name of a previously defined and added TERM object.

term-attribute

specifies an attribute describing a TERM object. A TERM attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET TERM command. Use any of the attributes listed for the SET TERM command.

RESET

changes an existing attribute of a TERM object to the default value.

If an attribute is not required and no default value exists, this option deletes the value set for the attribute.

If an attribute is required and no default value exists, the RESET option generates a syntax error and leaves the current attribute value unchanged.

term-keyword

specifies a single SET TERM command attribute keyword or several keywords separated by commas and enclosed in parentheses.

Consideration

If a TERM object is configured with the SET TERM INSPECT attribute set to ON and a file name, the ALTER TERM RESET command sets the INSPECT attribute to OFF and deletes the file name.

Errors

The following table lists the most common error that can occur during the processing of the ALTER TERM command:

This Message...	Is Displayed When...
1101 TOO MANY TERM ENTRIES	<p>You have tried to alter a TERM object but you have already added the maximum number of TERM entries (that is, 4095) supported by the PATHMON process.</p> <p>The command failed because the PATHMON process executes the ALTER TERM command by creating a new TERM entry with the altered values before deleting the existing TERM entry. When the maximum number of TERM entries already exists, the PATHMON process does not have the space to create a new TERM .</p> <p>To avoid this problem, define and add the maximum number of TERM entries allowed minus 1 (that is, 4094 TERMS).</p>

Examples

The following commands change various attribute values for the specified TERM objects:

```
ALTER TERM TERM-1,TMF OFF
ALTER TERM TERM-2,INITIAL IMFMENU
ALTER TERM-3,RESET AUTORESTART
ALTER TERM-4, TCP TCP-4, RESET (DISPLAY-PAGES,TYPE,PRINTER)
```

CONTROL TERM Commands

Use the CONTROL TERM commands to achieve synchronous upgrade for SCOBOL applications. The CONTROL TERM commands are:

- CONTROL TERM COUPLE
- CONTROL TERM DECOUPLE

These commands are used to:

- couple TERM to a SERVER in the specified PATHMON. All sends from this TERM to the given SERVER go to the specified PATHMON.
- decouple TERM to delete coupling information.

Note. The CONTROL TERM commands are supported by TS/MP 2.4 and are available only on systems running J06.05 and later J-series RVUs and H06.16 and later H-series RVUs.

```
CONTROL TERM  term name | ( term name, term name, ...)

      COUPLE SERVER server name WITH pathmon name
      CONTROL TERM term name | ( term name, term name, ...)
      DECOUPLE SERVER server name
```

term name

specifies the name of a TERM object for which couple or decouple operation must be performed.

server name

specifies the name of the server, which is either coupled with the specified PATHMON or which needs to be decoupled from the specified PATHMON.

pathmon name

is the PATHMON to which the requests divert after the CONTROL TERM COUPLE command.

Consideration

- When the CONTROL TERM COUPLE command is issued for a TERM, for all the requests targeted for the specified server, TCP (PATHTCP4) internally sets 9th bit of the *flags* parameter of the SERVERCLASS_SEND[L]_ procedure and changes the PATHMON name to the name specified in the CONTROL TERM COUPLE command. Therefore, all requests from the TERM for the specified server are always diverted to the specified PATHMON. TCP then forwards this

request to the access control server (ACS) infrastructure for delivering it to the appropriate server process.

- Use the CONTROL TERM DECOUPLE command to deliver the requests from the TERM for the specified server to any PATHMON in the domain.
- Specify the server and PATHMON names appropriately.
- The TERM must be in the STOPPED state for COUPLE request.
- The TERM can be in the STOPPED or RUNNING state for DECOUPLE request.

Examples

- The following command couples the TERM object named TERM1 with the PATHMON server PM1:

```
= CONTROL TERM TERM1 COUPLE SERVER SERVER1 WITH $PM1
```

- The following command decouples the TERM object named TERM1 for the PATHMON server PM1:

```
= CONTROL TERM TERM1 DECOUPLE SERVER SERVER1
```


DELETE TERM Command

Use the DELETE TERM command to remove a TERM object description from the PATHMON configuration file. A TERM object must be stopped before PATHCOM can delete it.

This command can be issued to configured and temporary TERM objects.

```
DELETE [ TERM ] { term-name
                  { ( term-name [, term-name ]... ) } }
```

term-name

specifies the name of a TERM object. Use either a single name or several names separated by commas and enclosed in parentheses.

Consideration

This command is usually issued only to configured TERM objects; temporary TERM objects created in response to a RUN PROGRAM command are usually terminated automatically by the PATHMON process when the task has completed.

Examples

The following command deletes the TERM object named TERM-1:

```
DELETE TERM TERM-1
```

The following command deletes two TERM objects:

```
DELETE (TERM3, TERM4)
```

INFO TERM Command

Use the INFO TERM command to display the attribute values defined in the PATHMON configuration file for a TERM object.

This command can be issued for configured and temporary TERM objects.

```
INFO [ / OUT list-file / ]

{ [ TERM ] term-name [ , OBEYFORM ] }
{ [ TERM ] (term-name [, term-name ]...) [ , OBEYFORM ] }
{ TERM * [ , option [ , option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
OBEYFORM
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

TERM *

displays the attribute values for all TERM objects described in the PATHMON configuration file. If *option* is specified, displays the attribute values for all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

```
RUNNING      NOT RUNNING
STOPPED      NOT STOPPED
SUSPENDED    NOT SUSPENDED
```

This option displays the attribute values for TERM objects in any state.

You can substitute SEL for STATE.

TCP *tcp-name*

displays information about all TERM objects controlled by the specified TCP.

OBEYFORM

displays the information in the format you would use to set up a PATHMON configuration file; each attribute is displayed as a syntactically correct PATHCOM SET command. PATHCOM adds a RESET TERM command before and an ADD TERM command after each TERM object description.

Considerations

- INFO TERM displays only the SET TERM attribute values that you explicitly defined for a TERM object. The attributes are displayed in alphabetic order. This command returns information about a TERM object only after the description has been added to the PATHMON configuration file.
- OBEYFORM is an INFO command option for the PATHMON, PATHWAY, TCP, TERM, SERVER, and PROGRAM objects. By using the OUT *list-file* option and issuing an INFO command for each object in succession, you can build a new cold start command file that reflects any modifications you have made to your PATHMON environment configuration. You must add the START PATHWAY command at the end of the OUT file before using the file to start a PATHMON environment.

Examples

The following command requests information for two TERM objects:

```
INFO (TERM-2,TERM-3)
```

The following command:

```
INFO TERM TERM-1, OBEYFORM
```

causes the PATHMON process to display the TERM object attributes in the following format:

```
RESET TERM
SET TERM AUTORESTART 0
SET TERM BREAK OFF
SET TERM DIAGNOSTIC ON
SET TERM ECHO ON
SET TERM FILE \*. $BOBE
SET TERM INITIAL MENU
SET TERM INSPECT OFF
SET TERM TCP TCPA
SET TERM TMF ON
ADD TERM TERM-1
```

The following command directs the information to a particular file:

```
INFO/OUT TERMFLE/TERM *
```

The following command displays attribute values for all TERM objects in the STOPPED state under the control of the TCP named TCPA:

```
INFO TERM *, STATE STOPPED, TCP TCPA
```

The following sequence of commands captures a configuration for a complete PATHMON environment (minus the START PATHWAY command) in the command file called NEWCONFIG:

```
=INFO/OUT NEWCONFIG/PATHMON, OBEYFORM
=INFO/OUT NEWCONFIG/PATHWAY, OBEYFORM
=INFO/OUT NEWCONFIG/TCP *, OBEYFORM
=INFO/OUT NEWCONFIG/TERM *, OBEYFORM
=INFO/OUT NEWCONFIG/SERVER *, OBEYFORM
=INFO/OUT NEWCONFIG/PROGRAM *, OBEYFORM
```

Note. For information about the PATHMON, PATHWAY, and SERVER objects listed in this example, see the *TS/MP System Management Manual*

INSPECT TERM Command

Use the INSPECT TERM command to invoke the HP *Inspect* symbolic debugger process to debug a SCREEN COBOL program running on a TERM object. This command establishes a breakpoint at the next program instruction; the program waits for commands from the Inspect process.

Before you use the INSPECT TERM command, the SET TCP INSPECT attribute must be set to ON.

This command can be issued for configured and temporary TERM objects.

```
INSPECT [ TERM ] term-name [ , FILE file-name ]
```

term-name

specifies the name of the TERM object on which the SCREEN COBOL program (the program to be debugged) is running.

FILE *file-name*

specifies the name of the Inspect command terminal (the terminal on which you enter the Inspect commands). The first character of the file name is a \$ (dollar sign) followed by one to six alphanumeric characters. The file name cannot exceed seven characters. You cannot specify a node name as part of the file name; the node is always the node on which the PATHMON process is currently running.

If *file-name* is omitted, the Inspect command terminal is one of the following, listed in order of priority:

1. The terminal named in the SET TERM INSPECT *file-name* attribute

Note. You can issue the INSPECT TERM command in situation 1 only if you had specified SET TERM INSPECT OFF and had at the same time specified a file name. You might do this to specify a default Inspect command terminal to use whenever you want to debug a program running on your terminal.

2. The terminal named in the SET TCP INSPECT *file-name* attribute
3. The home terminal for the TCP

Consideration

It is not necessary to specify SET TERM INSPECT ON before using this command.

Examples

The following commands invoke the INSPECT process to debug a program running on the specified TERM objects:

```
INSPECT TERM TERM-1  
INSPECT TERM-2, FILE $TERM6
```

RESET TERM Command

Use the RESET TERM command to change the values for TERM object attributes from the ones you defined with the SET TERM command to the defaults. This command does not change the attributes of a TERM object already added to the PATHMON configuration file.

This command can be issued only to configured TERM objects.

```
RESET TERM [ term-keyword [ , term-keyword ]... ]
```

term-keyword is:

AUTORESTART	FILE	TCLPROG
BREAK	INITIAL	TCP
DIAGNOSTIC	INSPECT	TMF
DISPLAY-PAGES	IOPROTOCOL	TRAILINGBLANKS
ECHO	MAXINPUTMSGs	TYPE
EXCLUSIVE	PRINTER	

term-keyword

specifies one or more attributes to be reset.

If you omit *term-keyword*, this command resets all attribute values to the defaults.

Considerations

- If you invoke the RESET TERM command after the SET TERM command but before the ADD TERM command, the attribute values you specify revert to the default values.
- Some required SET TERM command attributes have no default values. If you include a required attribute in the RESET TERM command, the attribute is set to a null value and must be set again before you add the TERM description to the PATHMON configuration file.

If an attribute is not required and does not have a default value, the RESET TERM command deletes the value for the attribute.

Examples

The following command resets all attribute values:

```
RESET TERM
```

The following commands reset the values for the specified attributes:

```
RESET TERM PRINTER
RESET TERM DIAGNOSTIC, TMF
```

RESUME TERM Command

Use the RESUME TERM command to restart a screen program that has been suspended with the SUSPEND TERM command. The RESUME TERM command changes the TERM state to RUNNING.

This command can be issued to configured and temporary TERM objects.

```
RESUME { [ TERM ] term-name
         { [ TERM ] ( term-name [ , term-name ]... )
         { TERM * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
TCP tcp-name
```

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

TERM *

restarts execution of all suspended TERM objects in the PATHMON environment. If *option* is specified, restarts all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

```
RUNNING      NOT RUNNING
STOPPED      NOT STOPPED
SUSPENDED    NOT SUSPENDED
```

This command restarts only those TERM objects in the SUSPENDED state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

TCP *tcp-name*

restarts all TERM objects controlled by the specified TCP.

Considerations

- After you enter a RESUME TERM command, the SCREEN COBOL program starts executing at the point where the TERM object was suspended. Screen recovery is performed before the program resumes normal execution.
- If a TERM object in TMF transaction mode was suspended for a programmatic reason and is in a restartable state, the RESUME TERM command initiates TERM activity by restarting the logical transaction at the BEGIN-TRANSACTION verb. When this happens, the special register RESTART-COUNTER is incremented by 1.
- If the TCP suspended a TERM object because of a fatal error condition, you might have to issue an ABORT TERM command before attempting to restart the TERM object.

Examples

The following command restarts the TERM object named TERM-1:

```
RESUME TERM TERM-1
```

The following command restarts two TERM objects:

```
RESUME (TERM-2 , TERM-3 )
```

The following command restarts all TERM objects in the SUSPENDED state:

```
RESUME TERM *
```

The following command restarts all TERM objects in the SUSPENDED state under the control of the TCP named TCPA:

```
RESUME TERM *, STATE SUSPENDED, TCP TCPA
```

SET TERM Command

Use the SET TERM command to define values for the TERM object attributes.

This command can be issued only to configured TERM objects.

```
SET TERM term-attribute [ , term-attribute ]...
```

term-attribute is:

```
FILE file-name
INITIAL program-unit-name
TCP tcp-name
AUTORESTART number
BREAK { ON | OFF }
DIAGNOSTIC { ON | OFF }
DISPLAY-PAGES number
ECHO { ON | OFF | CURRENT }
EXCLUSIVE { ON | OFF }
INSPECT { ON [ ( FILE file-name ) ] | OFF }
IOPROTOCOL { 0 | 1 }
LIKE term-name
MAXINPUTMSGs number
PRINTER { file-name [ IS-ATTACHED ] }
        { IS-ATTACHED }
TCLPROG file-name
TMF { ON | OFF }
TRAILINGBLANKS { ON | OFF }
TYPE term-class
```

term-attribute

The FILE, INITIAL, and TCP attributes have no defaults; you must specify them before adding a TERM definition. The rest of the attributes have default values.

For information about configuration limits, see [Appendix C, Configuration Limits and Defaults](#).

FILE *file-name*

specifies the name of a terminal or a process emulating a terminal by converting communications to the form required by a nonsupported terminal.

file-name must be represented in network form. It must begin with a \$ (dollar sign) followed by one to six alphanumeric characters, and the first character must be alphabetic. The file name cannot exceed a total of seven characters including the \$ symbol.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the *NonStop* TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET

PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

The following is an example of a file name for a process emulating a terminal:

```
\*. $PRO45 . #X25LA . TERM45
```

where \$PRO45 is the process file name and #X25LA is a communication line. If you define a process file name, you must also define a corresponding terminal type using the TYPE attribute.

This attribute is required to create a valid definition for a TERM object.

INITIAL *program-unit-name*

specifies the name of the SCREEN COBOL program unit that the TERM object enters on startup. *program-unit-name* is the program name specified in the PROGRAM-ID sentence of the identification division in a SCREEN COBOL program.

This attribute is required to create a valid definition for a TERM object.

TCP *tcp-name*

specifies the name of the TCP that controls the TERM object operations.

This attribute is required to create a valid definition for a TERM object.

AUTORESTART *number*

specifies the number of times that the PATHMON process attempts to restart the TERM object within a fixed 10-minute interval after an abnormal termination causes the TCP to place the TERM object in the SUSPEND-ABORT or a SUSPEND-RESUME state. This state does not result from a normal ABORT, STOP, or SUSPEND command.

number can be a value from 0 through 32,767. If you omit this attribute, the default is 0.

After an abnormal termination, the action caused by this option is equivalent to using an ABORT TERM command followed by the START TERM command.

AUTORESTART does not affect TERM objects during the interval between normal SUSPEND and RESUME operations, and does not apply to devices using programs that were started through the RUN PROGRAM command.

BREAK { ON | OFF }

specifies whether the TCP accepts the Break key function for its terminals running in conversational mode.

If a process is running as a terminal emulator, the value for this attribute always defaults to OFF.

If you omit this attribute, the default is OFF. This attribute is effective only for terminals running in conversational mode.

ON

directs the TCP to accept the Break key function. If the TCP is executing a SCREEN COBOL program in conversational mode, the terminal operator can press the Break key to prematurely terminate ACCEPT and DISPLAY statements. If the terminal operator presses the Break key while the TCP is executing a DISPLAY statement, the terminal output stops and the DISPLAY statement is terminated normally with the special register TERMINATION-SUBSTATUS set to 1. If the Break key is not used, TERMINATION-SUBSTATUS is set to 0 after the completion of a DISPLAY statement.

If the terminal operator presses the Break key while the TCP is executing an ACCEPT statement that is coded with an ESCAPE ON ABORT clause, the TCP stops accepting terminal input and terminates the ACCEPT statement without changes to the working storage section. Pressing the Break key has no effect on the TCP when the ACCEPT statement is coded without an ESCAPE ON ABORT clause. If the TCP is not executing either of the preceding statements, pressing the Break key has no effect on the TCP process.

OFF

directs the TCP to ignore the Break key function.

DIAGNOSTIC { ON | OFF }

specifies whether diagnostic screens are displayed to inform the terminal operator when an error condition or termination occurs.

The SCREEN COBOL special register DIAGNOSTIC-ALLOWED is initialized to YES or NO according to the value of this attribute. Screen recovery is invoked following the display of a diagnostic screen.

If you omit this attribute, the default is ON.

ON

displays diagnostic screens. The first DISPLAY BASE statement in the SCREEN COBOL program must have executed before this option can cause a diagnostic screen to appear.

OFF

does not display diagnostic screens.

DISPLAY-PAGES *number*

specifies the depth of the terminal's screen caching. DISPLAY-PAGES determines the maximum number of screen displays that are stored in the terminal's memory.

If you omit this attribute, the default is used. If you specify a value greater than the defined maximum, the defined maximum is used.

Terminal Type	Default Value	Defined Minimum	Defined Maximum
T16-6540	7	1	16*
T16-6530WP	7	1	8*
T16-6530	7	1**	8*
T16-6520	3	1**	3
IBM-3270	1	1	1
TS530	8	1	8*

* This is the absolute maximum allowed for the TCP. However, if the terminal or terminal emulator has less memory, the working value is negotiated downward.

** Although the PATHMON process allows you to set the number of pages to 1, some terminals or terminal emulators do not honor that setting. For example, the T16-6520 terminal always sets its number of display pages to 3, regardless of the user setting.

You select the appropriate DISPLAY-PAGES value based on the following application-dependent information:

- The maximum number of fields per screen that are declared in the SCREEN COBOL program
- The average number of fields per screen the terminal type can handle for the particular setting of the DISPLAY-PAGES option

Terminal memory is divided into two separate areas: the screen image area and the field attribute area. The DISPLAY-PAGES attribute determines how much terminal memory is used for screen images as opposed to how much is used for field attributes. The more memory assigned to one, the less is available to the other. Therefore, if your application has many fields per screen, you should set DISPLAY-PAGES to a small value to leave room for the field attributes. If you specify too many display pages, you might receive the terminal error ALL FIELDS USED.

For detailed information on how to select the appropriate value for the DISPLAY-PAGES attribute, see [Appendix F, Setting the DISPLAY-PAGES Parameter](#).

ECHO { ON | OFF | CURRENT }

overrides the terminal echo mode (that is, the mode configured for a terminal at system generation time) so that character input appears correctly on the terminal. The ECHO attribute supports terminals running in conversational or intelligent mode.

If you omit this attribute, the default is ON.

ON

directs input to appear on the terminal screen.

OFF

corrects the display by suppressing the duplicate characters (if a terminal is configured so that it displays input characters). If you define this option for the supported terminal types T16-6520, T16-6530, and T16-6540, no input characters appear on the terminal screen.

CURRENT

directs the TCP to obtain the operating mode of the terminal from the operating system and to use that mode when it opens the terminal. If you specify this option for a process that is running as a terminal emulator, this attribute value defaults to ON.

EXCLUSIVE { ON | OFF }

specifies whether a TCP should open a terminal for exclusive or shared access.

If you omit this attribute, the default is OFF.

ON

directs a TCP to open the terminal for exclusive access. The TCP cannot allow simultaneous access of any kind to the terminal by any other process.

OFF

directs a TCP to open the terminal for shared access. The TCP allows simultaneous read or write access to the terminal by other processes.

INSPECT { ON [(FILE *file-name*)] | OFF }

specifies whether a SCREEN COBOL program running on this TERM object starts the Inspect process when the program begins execution. The SET TCP INSPECT attribute must be set to ON.

If you omit this attribute, the default is OFF.

ON

starts the Inspect process when the program begins and positions the program at a breakpoint before the first program instruction executes.

FILE *file-name*

specifies the name of the Inspect command terminal.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the NonStop TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

If you omit *file-name* and the Inspect command terminal is not named in the SET TCP INSPECT attribute, the default is the home terminal of the TCP.

OFF

does not start the Inspect process when the program begins.

IOPROTOCOL { 0 | 1 }

specifies the protocol the TCP follows when a SCREEN COBOL SEND MESSAGE statement completes with a timeout or an unsolicited message.

If you omit this attribute, the default is 0.

0

directs a TCP to issue a CANCEL procedure to cancel a SEND MESSAGE I/O. The I/O is cancelled because an unsolicited message arrives or a SEND MESSAGE TIMEOUT occurs.

1

directs a TCP to use a CONTROL-26 protocol to cancel a SEND MESSAGE I/O. The I/O is cancelled because an unsolicited message arrives or a SEND MESSAGE TIMEOUT occurs.

If IOPROTOCOL 1 is specified, the front-end process must be written according to rules described in the *TS/MP Pathsend and Server Programming Manual*. Front-end processes can use this attribute to avoid loss of data.

LIKE *term-name*

sets the attribute values for the TERM object to those of a previously added TERM object.

MAXINPUTMSGGS *number*

specifies the maximum number of unsolicited messages that the TCP queues for the requester associated with the TERM object. If this attribute is exceeded, the TCP returns a TERM^QUEUE^FULL error to the originator of the unsolicited message.

number must be a value from 0 through 2045. If you omit this attribute, the default is 0.

PRINTER

specifies the printer to be used for the SCREEN COBOL PRINT SCREEN verb.

An attached printer either physically plugs into the terminal (for example, a 6520 or 6530 terminal) or connects to the same line controller (for example, an IBM 3270 compatible terminal using AM3270) as the terminal on which the SCREEN COBOL program runs.

file-name

specifies the file name for the print device. You usually use *file-name* when the printer is not attached or when multiple attached printers are connected to a line controller. This attribute overrides the current value configured for the PRINTER attribute.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the NonStop TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

The specified file name is expanded and then placed in the SCREEN COBOL special register TERMINAL-PRINTER when the program starts running on a terminal. While a screen program runs, it can modify the file name in its TERMINAL-PRINTER register and then use the new file name.

If you omit *file-name*, the SCREEN COBOL TERMINAL-PRINTER special register is set to SPACES (that is, ASCII blanks).

If a printer is not attached, the SCREEN COBOL verb PRINT-SCREEN obtains the printer's file name from the special register TERMINAL-PRINTER.

IS-ATTACHED

directs the copy of the screen image to a printer that is either physically attached to the terminal or on the same line controller as the terminal on which the program runs. IS-ATTACHED is one of the following:

T16-6530 Directs copy to a printer attached directly to the terminal.

T16-6520 Directs copy to a printer attached directly to the terminal.

IBM-3270 Directs copy to the device that is attached to the same control unit as the terminal.

If you specify PRINTER IS-ATTACHED for a 6520, 6530, or 6540 terminal, the SCREEN COBOL special register TERMINAL-PRINTER is ignored.

TCLPROG *file-name*

specifies the name of the SCREEN COBOL object library file that contains the screen programs for this TERM object. The TCP checks this file name for the requested program unit before checking the SET TCP TCLPROG file name. If you omit this option, the TCP uses the file defined for the SET TCP TCLPROG attribute.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the NonStop TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

In a production environment, it is better not to specify a TCLPROG filename for TERM (and PROGRAM) objects. If you specify a TCLPROG filename, the TCP checks the file every time a SCREEN COBOL program unit is requested.

TMF { ON | OFF }

specifies whether the TMF subsystem runs with this TERM object. This option is intended to be used only as a debugging aid. For more information about this attribute, see [Appendix E, Setting TMF Parameters](#).

If you omit this attribute, the default is ON.

ON

invokes TMF auditing when necessary. The ABORT-TRANSACTION, BEGIN-TRANSACTION, END-TRANSACTION, and RESTART-TRANSACTION verbs perform the functions described in the *Pathway/iTS SCREEN COBOL Reference Manual*.

OFF

prevents TMF auditing. The TRANSACTION-ID special register is set to SPACES after the BEGIN-TRANSACTION verb executes.

TRAILINGBLANKS { ON | OFF }

specifies whether the TCP suppresses the display of trailing ASCII blank characters in output lines sent to terminals by the SCREEN COBOL DISPLAY and SEND MESSAGE verbs.

If you omit this attribute, the default is ON.

ON

directs the TCP not to suppress trailing ASCII blanks.

OFF

directs the TCP to suppress trailing ASCII blanks.

This attribute affects terminals defined as TYPE CONVERSATIONAL and affects programs defined as TYPE INTELLIGENT that use delimited format messages. This attribute has no effect on VARYING1 and VARYING2 message formats.

TYPE *term-class*

specifies the type of device on which the TERM object runs. Specify *term-class* in the format:

term-type:term-subtype

The possible values are:

1: <i>n</i>	or	IBM-3270: <i>n</i>	IBM 3270 terminal
3:0	or	T16-6520:0	6520 terminal
4:0	or	T16-6530:0	6530 terminal
5:0	or	CONVERSATIONAL	Conversational-mode terminal
6:0	or	T16-6540:0	6540 terminal emulating a 6530 terminal
7: <i>n</i>	or	INTELLIGENT: <i>n</i>	Intelligent device or web client
8:0	or	T16-6530WP:0	6530WP terminal

For intelligent devices, *term-type* must be either 7 or INTELLIGENT, and *term-subtype*, which identifies the I/O protocol, must be one of the following:

- 0 WRITEREAD I/O protocol; write to device and wait for reply (default) in conversational mode
- 1 WRITE and READ I/O protocol; write to device and then read from device in block mode
- 2 WRITEREAD I/O protocol; write to device and wait for reply in block mode

For IBM 3270 terminals, *term-type* must be either 1 or IBM-3270, and *term-subtype*, which specifies the screen display, must be one of the following:

<i>term-subtype</i>	Screen Size	Model
1	480, 12 x 40	IBM 3277 M1
2	1920, 24 x 80	IBM 3277 M2
3	2560, 24 x 80, alternate 32 x 80	IBM 3278 M3
4	3440, 24 x 80, alternate 43 x 80	IBM 3278 M4
5	960, 12 x 40, alternate 12 x 80	IBM 3278 M1
6	3564, 24 x 80, alternate 27 x 132	IBM 3278 M5

For information about using IBM 3270 terminals in a Pathway environment, see the *Pathway/iTS TCP and Terminal Programming Guide*.

If the TERMINAL IS clause of the running SCREEN COBOL program is not present, the *term-type* value specified in this attribute is used. If the TERMINAL IS clause is not present and this attribute does not specify a *term-type* value, the TCP determines the device type from a call to the file-system DEVICEINFO procedure.

If the TERMINAL IS clause of the running SCREEN COBOL program does not specify a value for *term-subtype*, the *term-subtype* value specified in this attribute is used. If the TERMINAL IS clause is not present and this attribute does

not specify a *term-subtype* value, the TCP determines the device type from a call to the file-system DEVICEINFO procedure.

If a program unit for an intelligent device does not specify a *term-subtype* value in the TERMINAL IS clause or a *term-subtype* value is not specified in this attribute, a default value is used rather than the value obtained from the file-system DEVICEINFO procedure.

Considerations

- If a SCREEN COBOL program is running on an incompatible terminal type, the results can be unpredictable depending on the terminal type. For example, the results are unpredictable when a program compiled for an IBM 3270 terminal is executed on a 6530 terminal (although the TCP does not prohibit this execution). Unpredictable results can also occur when a program compiled for a 6520, 6530, or conversational-mode terminal is executed on a 6540 terminal that is defined as a 6530 terminal emulator (*term-type* of 6:0).
- If you repeat a SET TERM command with a different attribute value than the one entered with an earlier SET TERM command, PATHCOM uses the last value entered for the TERM attribute.
- The TCP opens terminals with a syncdepth of 1.

Examples

The following SET TERM commands define a subset of the TERM attributes:

```
SET TERM AUTORESTART 4
SET TERM DISPLAY-PAGES 6
SET TERM FILE $T6530
SET TERM INITIAL MAINMENU
SET TERM TCP TCP-1
SET TERM INSPECT ON (FILE $TERMY)
SET TERM TYPE 4:0
```

SHOW TERM Command

Use the SHOW TERM command to display a subset of the TERM attributes in alphabetic order. This command displays the attribute values for a TERM object not yet added to the PATHMON configuration file.

This command can be issued only to configured TERM objects.

```
SHOW [ / OUT list-file / ] TERM
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

Considerations

- When PATHCOM first starts and after a RESET TERM command executes, the SHOW TERM command displays only the required SET TERM attributes and a subset of the optional attributes that have default values. A question mark appears for the required attribute values that you must set before adding the TERM object.
- SHOW TERM displays all of the attribute values that you explicitly set with the SET TERM command.
- For attributes that include *Guardian* operating environment file names, SHOW TERM displays the node-name part of the file name as either `\nodename` or `*`. The `\nodename` form is used if the node was explicitly specified by the user. The `*` form is used if the attribute was specified to default to the node where the PATHMON process is running.
- After an ADD TERM command, the SHOW TERM command continues to display the attribute values held in PATHCOM for the previously added TERM object. Use the RESET TERM command to change all of the values to their defaults, or use the SET TERM command to replace specific attribute values.

Examples

If you issue the SHOW TERM command before setting the TERM object attributes, the following information is displayed:

```
TERM
  AUTORESTART 0
  BREAK OFF
  DIAGNOSTIC ON
  ECHO ON
  FILE ?
  INITIAL ?
  INSPECT OFF
```

TCP ?
TMF ON

The following command directs the screen displays to a particular file:

SHOW/OUT TERMFLE/TERM

START TERM Command

Use the START TERM command to direct the TCP to begin executing a SCREEN COBOL program on one or more designated TERM objects. The TCP that controls the TERM objects must be running before you issue this command.

This command can be issued only to configured TERM objects.

```
START { [ TERM ] term-name
        [ , INITIAL program-unit-name ] }
      { [ TERM ] ( term-name [ , term-name ]... )
        [ , INITIAL program-unit-name ] }
      { TERM * [ , option [ , option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
INITIAL program-unit-name
```

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

INITIAL *program-unit-name*

specifies the name of the SCREEN COBOL program unit that runs on the TERM object when it starts.

If you omit this option, the default is the initial program unit established by the SET TERM command.

TERM *

starts all TERM objects in the PATHMON environment controlled by TCPs running in the same PATHMON environment. If *option* is specified, starts all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

RUNNING	NOT RUNNING
STOPPED	NOT STOPPED
SUSPENDED	NOT SUSPENDED

This command starts only those TERM objects in the STOPPED state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

TCP *tcp-name*

starts all TERM objects controlled by the specified TCP.

Consideration

The START TERM is valid only for configured TERM objects. Temporary TERM objects are started when the RUN PROGRAM command executes.

Examples

The following commands start only the specified TERM objects:

```
START TERM TERM-1
START TERM-2
START (TERM-3, TERM-4)
START TERM-5, INITIAL IMFMENU
```

The following command starts all TERM objects in the STOPPED state that have TCPs running:

```
START TERM *
```

The following command starts all TERM objects in the STOPPED state under the control of the TCP named TCPA:

```
START TERM *, STATE STOPPED, TCP TCPA
```

STATS TERM Command

Use the STATS TERM command to display resource-usage and performance statistics, including response time information, collected by the TCP. These statistics provide information about TERM object operations. The STATS attributes of the SET TCP command control the gathering of these statistics.

This command can be issued for configured and temporary TERM objects.

```
STATS [ / OUT list-file / ]

  { [ TERM ] term-name
    [ , term-attribute [ , term-attribute ]... ]
  }
  { [ TERM ] ( term-name [ , term-name ]... )
    [ , term-attribute [ , term-attribute ]... ]
  }
  { TERM * [ , option [ , option ] ... ]
  }
```

term-attribute is:

```
COUNT number
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET
```

option is:

```
STATE state-type
TCP tcp-name
COUNT number
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

term-name

specifies one or more TERM objects. *term-name* can be either the name of a single TERM object or several TERM object names separated by commas and enclosed in parentheses.

COUNT *number*

specifies the number of times that the STATS command repeats. You can stop the repeating of the command by pressing the Break key.

FREQTABLE

generates a frequency distribution table containing statistics for each terminal. The frequency distribution table is not generated—even if you specify the FREQTABLE option—under the following conditions:

- There have been fewer than 50 response time measurements collected at the time of the STATS request.
- At the fiftieth measurement, the size of the time interval calculated is less than 0.01 second.

For more information about the FREQTABLE option, including a sample display, see [Section 5, Tuning Your System Using Statistics](#).

INTERVAL *number*

specifies the time period between the statistics displays, incremented in hours, minutes, or seconds.

RESET

sets the counters used for the measurements back to zero.

TERM *

displays statistics for all TERM objects described in the PATHMON configuration file. If *option* is specified, displays statistics for all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

RUNNING	NOT RUNNING
STOPPED	NOT STOPPED
SUSPENDED	NOT SUSPENDED

This command displays statistics only for those TERM objects in the RUNNING or SUSPENDED state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

TCP *tcp-name*

displays statistics for all TERM objects controlled by the specified TCP.

Considerations

- If the SET TCP STATS attribute is set to OFF, the TCP does not gather information for this command. Use the CONTROL TCP STATS ON command to dynamically enable the TCP statistics collection mechanism.
- For more information about the STATS TERM command display, see [Section 5, Tuning Your System Using Statistics](#).

Examples

The following command displays the statistical information gathered for the TERM object named TERM-1 for a 90-second interval:

```
STATS TERM-1, INTERVAL 90 SECS
```

The following command directs nonaccumulating statistics for all terminals to a list file named STATFLE every 30 minutes:

```
STATS /OUT STATFLE/ TERM *,RESET,INTERVAL 30 MINS
```

The following command displays statistics for all TERM objects in the RUNNING state under the control of the TCP named TCPA:

```
STATS TERM *, STATE RUNNING, TCP TCPA
```

STATUS TERM Command

Use the STATUS TERM command to display the current status of a TERM object.

This command can be issued for configured and temporary TERM objects.

```
STATUS [ / OUT list-file / ]

{ [ TERM ] term-name [ , DETAIL ] }
{ [ TERM ] ( term-name [ , term-name ]... ) [ , DETAIL ] }
{ TERM * [ , option [ , option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
DETAIL
```

OUT *list-file*

directs output to the file that you specify; this could be a DEFINE name. If you omit this option, the PATHMON process writes the output to the PATHCOM list file; this is typically the home terminal.

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

DETAIL

includes status information on the TERM objects and the names of the program units running on them.

TERM *

displays the status of all TERM objects described in the PATHMON configuration file. If *option* is specified, displays the status of all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

```
RUNNING      NOT RUNNING
STOPPED      NOT STOPPED
SUSPENDED    NOT SUSPENDED
```

This option displays the status of TERM objects in any state.

You can substitute SEL for STATE.

TCP *tcp-name*

displays status information for all TERM objects controlled by the specified TCP.

Display Format Without DETAIL

The format of the display returned by the STATUS TERM command without the DETAIL option is shown below.

TERM	STATE	ERROR	INFO	TCP	FILE
<i>term-name</i>	<i>state</i>	<i>pw-error</i>	<i>info</i>	<i>TCP-name</i>	<i>file-name</i>
.
.

The fields in this display are as follows:

Display Field	Description
TERM	Name of the TERM object
STATE	Current state of the TERM object. Possible values are: STOPPED Terminal is stopped RUNNING Terminal is running SUSPENDED Terminal is suspended
ERROR	PATHMON environment error number
INFO	Additional information regarding the error number. See the error messages listed in Section 13, TCP Messages (Numbers 3000-3999) .
TCP	Name of the controlling TCP
FILE	Name of the terminal

Display Format With DETAIL

The format of the display returned by the STATUS TERM command with the DETAIL option is shown below.

TERM	STATE	ERROR	INFO	TCP	FILE
<i>term-name</i>	<i>state</i>	<i>pw-error</i>	<i>info</i>	<i>TCP-name</i>	<i>file-name</i>
.
.
WAIT	PENDING	ACCEPT	STOPMODE	TRANSMODE	
<i>wait-cause</i>	<i>pend-cause</i>	<i>accept</i>	<i>stopmode</i>	<i>transmode</i>	
				[RECOVERING]	
.
.
PU-FILE		<i>TCL-program-file-name</i>			
PU-NAME		<i>program-unit</i>			
PU-VRSN		<i>program-unit-version</i>			
INST-ADDR		<i>instruction-address</i>			
INST-CODE		<i>instruction-code</i>			
[CHAR-SET		<i>character-set-type</i>]	
[TRANS-ID		<i>transaction-identifier</i>]	
[TRANS-RESTARTS		<i>transaction-restarts</i>]	
[SERVER		<i>server-name</i>]	
[COUPLING		<i>server-name pathmon-name</i>]	

The first additional line is similar to that shown for the STATUS TCP DETAIL display in [Section 9, Terminal Control Process \(TCP\) Commands](#).

The TERM, STATE, ERROR, INFO, TCP, and FILE fields have the same meanings as for the display without the DETAIL option. The additional fields returned in the detailed display are as follows:

Display Field	Description
PU-FILE	Name of the currently executing SCREEN COBOL program file from the SET TERM or the SET TCP TCLPROG attribute
PU-NAME	Name of the currently executing program unit
PU-VRSN	Version of the program unit indicated by numbers. The first version is assigned number 1. The version number increments by 1 for each version thereafter.
INST-ADDR	Octal offset from the base of the program unit of the currently executing instruction
INST-CODE	Name of the currently executing instruction
CHAR-SET	USASCII or foreign character set supported for the 6530, 6530WP, and 6540 terminals. The character set is specified in the OBJECT-COMPUTER paragraph of the associated SCREEN COBOL program.

Display Field	Description
TRANS-ID	Identifier assigned to a transaction by the TMF subsystem
TRANS-RESTARTS	Number of times a transaction has been restarted
SERVER	Name of the server class the terminal is communicating with or waiting for

The STATUS TERM command now displays the coupling information when issued with the DETAIL option. The following example shows the command output with coupling information:

```
\NSSYS.$VOL.SUBVOL 11> PATHCOM
$X30Q: PATHCOM - T0845H02 - (30APR09)
(C)2008-2009 Hewlett Packard Development Company, L.P.
= [ A domain %dom is configured with $PM1, $PM2 as it is
= [ member PATHMONs. Term TERM1 is configured under $PM1.
= [ Server SERVER1 is configured under both $PM1 and $PM2.
=

= CONTROL TERM TERM1, COUPLE SERVER SERVER1 WITH $PM1
$X30Q: TERM1 COUPLED WITH $PM1.SERVER1
= STATUS TERM TERM1, DETAIL
TERM      STATE      ERROR  INFO      TCP          FILE
TERM1     STOPPED                                TCP-T001     \NSSYS.$ZTN0.#PTG1ABK
          COUPLING    SERVER1      $PM1
```

Display Format With DETAIL During Shutdown

When a PATHMON environment is in the process of shutting down, it is possible to issue the STATUS TERM command with the DETAIL option at a time when the specified terminal is stopped but the TCP has not yet reported the stopped state to the PATHMON process. In this case, the returned screen displays the following:

- The TERM, the TCP, and the FILE fields, which are supplied by the PATHMON process, contain current information.
- The STATE field, which is supplied by the PATHMON process, always contains RUNNING.
- The WAIT field, which is supplied by the TCP, always contains SUSP-RESTART; this indicates that the terminal is stopped.
- All other fields are initialized to 0.

[Example 10-1](#) shows an example screen returned for a terminal named TERM-1 that has been stopped by the TCP but the stopped state has not yet been reported to the PATHMON process.

Example 10-1. Example STATUS TERM Display With DETAIL During Shutdown

TERM	STATE	ERROR	INFO	TCP	FILE
TERM-1	RUNNING			TCP-1	\MARS.\$PWY.#FILE1
WAIT	PENDING	ACCEPT	STOPMODE	TRANSMODE	
SUSP-RESTART					
PU-FILE					
PU-NAME					
PU-VRSN		0			
INST-ADDR		%000000			
INST-CODE		%000000			

Consideration

In a production environment with a large number of terminals, it is better not to use the TERM * option to return status information on all TERM objects in the PATHMON environment. Instead, use an explicit TERM object name to return status information on a particular TERM object, or use the STATE option to request status information on TERM objects in a particular state.

Examples

The following commands request status information on a specified TERM object(s):

```
STATUS TERM TERM-1
STATUS/OUT STATFLE/TERM (TERM-2,TERM-3)
STATUS TERM TERM-4 DETAIL
STATUS TERM *, DETAIL
STATUS/OUT STATFLE/TERM *
```

The following command requests status information for all TERM objects in the SUSPENDED state under the control of the TCP named TCPA:

```
STATUS TERM *, STATE SUSPENDED, TCP TCPA
```

STOP TERM Command

Use the STOP TERM command to stop a TERM object. The TCP stops a TERM object as soon as the TERM object reaches a qualified state. A qualified state exists when the following three conditions are met:

- The TERM object has reached a SCREEN COBOL ACCEPT statement.
- The SCREEN COBOL special register STOP-MODE is set to zero.
- The TERM object (if running under TMF auditing) is not in TMF transaction mode.

This command can be issued to configured and temporary TERM objects.

```
STOP { [ TERM ] term-name
      { [ TERM ] ( term-name [ , term-name ]... )
      { TERM * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
TCP tcp-name
```

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

TERM *

stops all TERM objects in the PATHMON environment. If *option* is specified, stops all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

```
RUNNING      NOT RUNNING
STOPPED      NOT STOPPED
SUSPENDED    NOT SUSPENDED
```

This command stops only those TERM objects in the RUNNING or SUSPENDED state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

TCP *tcp-name*

stops all TERM objects controlled by the specified TCP.

Consideration

- This command is usually issued only to configured TERM objects; temporary TERM objects created in response to a RUN PROGRAM command are usually terminated automatically by the PATHMON process when the task has completed.
- The coupling information is lost after issuing a STOP command.

Examples

The following commands stop the specified TERM objects:

```
STOP TERM-1  
STOP TERM TERM-2  
STOP TERM (TERM-3, TERM-4)
```

The following command stops all TERM objects in the RUNNING state under the control of the TCP named TCPA:

```
STOP TERM *, STATE RUNNING, TCP TCPA
```

The following command stops all TERM objects in the PATHMON environment:

```
STOP TERM *
```

SUSPEND TERM Command

Use the SUSPEND TERM command to direct the TCP to temporarily suspend execution of the SCREEN COBOL program. This command can be used for debugging when it is important to preserve the TERM object in its most recent state. The RESUME TERM command releases the suspension.

A TERM object is suspended as soon as it reaches a qualified state, which exists when the following three conditions are met:

- The TERM has reached a SCREEN COBOL ACCEPT statement.
- The SCREEN COBOL special register STOP-MODE is set to zero.
- The TERM (if running under TMF auditing) is not in TMF transaction mode.

This command can be issued to configured and temporary TERM objects.

```
SUSPEND { [ TERM ] term-name [!] }
        { [ TERM ] ( term-name [ , term-name ]... ) }
        { TERM * [ option [ option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
!
```

term-name

specifies the name of a TERM object. You can use either a single TERM name or several TERM names separated by commas and enclosed in parentheses.

!

immediately suspends TERM objects at any ACCEPT statement; the STOP-MODE value is disregarded. If a TERM object is in TMF transaction mode, the transaction is backed out before the TERM object is suspended.

TERM *

suspends execution of all TERM objects in the PATHMON environment. If *option* is specified, suspends all TERM objects meeting the criteria specified.

STATE *state-type*

specifies the state of the multiple TERM objects. Possible values are:

```
RUNNING      NOT RUNNING
STOPPED      NOT STOPPED
SUSPENDED    NOT SUSPENDED
```

This command suspends only those TERM objects in the RUNNING state. If you specify any other state, the system redisplay the command prompt; it does not display any output or generate an error message.

You can substitute SEL for STATE.

TCP *tcp-name*

suspends all TERM objects controlled by the specified TCP.

Examples

The following commands suspend the specified TERM objects:

```
SUSPEND TERM TERM-1  
SUSPEND (TERM-2, TERM-3)
```

The following command suspends all TERM objects in the RUNNING state under the control of the TCP named TCPA:

```
SUSPEND TERM *, STATE RUNNING, TCP TCPA
```

The following command immediately suspends all TERM objects in the PATHMON environment:

```
SUSPEND TERM * !
```

Wild Card Support for TERM Commands

The following TERM commands in PATHCOM allow management and operation of all TERM objects configured under a PATHMON environment by using the wild card "*":

- ABORT
- INFO
- RESUME
- START
- STATS
- STATUS
- STOP
- SUSPEND

All these commands can be applied to a subset of configured TERM objects.

Note. The above mentioned wild card support is applicable only to TS/MP 2.3 PATHMON and PATHCOM. For more information on TS/MP 2.3, see the *TS/MP Release Supplement Manual*.

Example

The following command displays the status information for all terms configured under the PATHMON environment whose name starts with TER:

```
STATUS TERM TER*
```

The system displays this output:

TERM	STATE	ERROR	INFO	TCP	FILE
TERM1	RUNNING			TCP1	\ACS.\$ZTNT.#PTTAADV
TERM2	RUNNING			TCP1	\ACS.\$ZTNT.#PTTAADV
TERM3	RUNNING			TCP1	\ACS.\$ZTNT.#PTTAADV
TERMA	RUNNING			TCP1	\ACS.\$ZTNT.#PTTAADV

Synchronous Upgrade for SCOBOL Applications

The online upgrade feature is useful only if the old and new versions are compatible; a new server can process requests from the older requester and vice versa. This is often not true for an upgraded SCOBOL application. The objective of the synchronous upgrade feature is to achieve zero downtime for such cases.

The following CONTROL commands are introduced for the synchronous upgrade:

- CONTROL DOMAIN PARTITION
- CONTROL DOMAIN UNDO PARTITION
- CONTROL TERM COUPLE
- CONTROL TERM DECOUPLE

For information on CONTROL DOMAIN commands, see the *TS/MP Release Supplement Manual*. For information on CONTROL TERM commands, see [CONTROL TERM Commands](#) on page 10-7.

The STATUS TERM command is also enhanced to support this feature.

Note. The synchronous upgrade feature is supported by TS/MP 2.4 and is available only on systems running J06.05 and later J-series RVUs and H06.16 and later H-series RVUs.

Procedure for Synchronous Upgrade

Perform the following steps to achieve synchronous upgrade for SCOBOL applications:

1. Select a PATHMON server that must be upgraded within the domain.
2. Issue the CONTROL DOMAIN PARTITION command for the selected PATHMON server (\$PM1.SRV) on the domain:

```
PDMI 1>> OPEN %dom
PATHMON : \NSSYS.$PM1
PATHMON : \NSSYS.$PM2
```

```
PDMI 2>> CONTROL DOMAIN PARTITION PATHMON $PM1 SERVER SRV
%dom PARTITIONED FOR $PM1.SRV
```

3. Upgrade the selected server (\$PM1.SRV) from the selected PATHMON using a combination of FREEZE, STOP, ALTER, THAW, and START commands. In most cases, the server will have a new executable, which is not compatible with the older TERMS:

```
PDMI 3>> OPEN $PM1
PATHMON : \NSSYS.$PM1
```

```
PDMI 4>> FREEZE SRV; STOP SRV
PATHMON : \NSSYS.$PM1
$Z2L0: SERVER SRV, FROZEN
STOP SRV
PATHMON : \NSSYS.$PM1
$Z2L0: SERVER SRV, STOPPED
```

```
PDMI 5>> ALTER SRV, PROGRAM newexe
PATHMON : \NSSYS.$PM1
```

```
PDMI 6>> THAW SRV; START SRV
PATHMON : \NSSYS.$PM1
$Z2L0: SERVER SRV, THAWED
START SRV
PATHMON : \NSSYS.$PM1
$Z2L0: SERVER SRV, STARTED
```

Follow steps 4 to 11 to alter TCP or TERMS or both:

4. Stop and alter the TERMS associated with the TCP that need to be upgraded.
5. Stop the TCP that needs to be altered.
6. Alter the TCP.
7. Bring the TCP up again.
8. Issue the CONTROL TERM COUPLE command for the TERMS that need to be upgraded, with the upgraded PATHMON server (\$PM1.SRV):

```
PDMI 7>> CONTROL TERM TERM1 COUPLE SERVER SRV WITH $PM1
$Z2L0: TERM1 COUPLED WITH $PM1.SRV
```

All future requests from these TERMS are sent to the upgraded PATHMON server (\$PM1 .SRV) with 9th bit of `flag` parameter of the `SERVERCLASS_SENDR_` procedure.

9. Start TERMS. Because of the prior `CONTROL` command, TERMS will send the request to the already upgraded PATHMON server.

```
PDMI 8>> START TERM1
$Z2L0: TERM TERM1, STARTED
```

All the terms send requests to the upgraded PATHMON server (\$PM1 .SRV). As a result, you might experience performance degradation for some time.

10. Upgrade the server in the remaining PATHMONs in the domain using a combination of `FREEZE`, `STOP`, `ALTER`, `THAW`, and `START` commands.
11. Issue the `CONTROL TERM DECOUPLE` command for all the upgraded TERMS. The requests go to any server (\$* .SRV) within the domain, except \$PM1 .SRV.

```
PDMI 9>> CONTROL TERM TERM1 DECOUPLE SERVER SRV
$Z2L1: TERM1 DECOUPLED FOR SRV
```

12. Issue the `CONTROL DOMAIN UNDO PARTITION` command to remove the partition from the earlier server (\$PM1 .SRV) . As a result, the original performance levels are resumed.

```
PDMI 10>> CONTROL DOMAIN UNDO PARTITION SERVER SRV
%dom PARTITION REMOVED FOR SRV
```

11 PROGRAM Commands

This section describes the PATHCOM commands that define and control PROGRAM objects; the commands are listed in alphabetical order. The PROGRAM commands are as follows:

To Perform the Following Tasks...

Define and remove PROGRAM objects

Control PROGRAM objects

Change PROGRAM object attributes

Obtain information about PROGRAM objects

Use These PATHCOM Commands...

SET PROGRAM

ADD PROGRAM

DELETE PROGRAM

RUN PROGRAM

ALTER PROGRAM

RESET PROGRAM

INFO PROGRAM

SHOW PROGRAM

These commands define and control templates used by the PATHMON process when creating temporary TERM objects in response to a RUN PROGRAM command. The templates define tasks that enable TCPs to run screen programs temporarily on one or more devices. With these commands you can do the following:

- Set and modify PROGRAM attributes
- Add and delete PROGRAM templates
- Run the initial screen program defined in the PROGRAM template and add and start a temporary TERM object
- Display PROGRAM attributes

ADD PROGRAM Command

Use the ADD PROGRAM command to enter the name and the description of a PROGRAM object into the PATHMON configuration file. Use this command after defining a PROGRAM object with the SET PROGRAM command.

```
ADD PROGRAM program-name [ , program-attribute ]...
```

program-name

specifies the PROGRAM object name. *program-name* can have from 1 to 15 alphanumeric or hyphen characters and must start with a letter, be unique within the PATHMON environment, and not be a PATHCOM reserved word.

program-attribute

specifies an attribute describing a PROGRAM object. A PROGRAM object attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET PROGRAM command. Use any of the attributes listed for the SET PROGRAM command.

Consideration

To use the ADD PROGRAM command, you must have a licensed copy of the HP NonStop Pathway/iTS product on your system; otherwise, the PATHMON process returns an error.

Examples

The following examples add program objects to the PATHMON configuration file:

```
ADD PROGRAM PROG-1  
ADD PROGRAM PROG-2, TCP TCP-1, TYPE T16-6530 (INITIAL MENU)
```


ALTER PROGRAM Command

Use the ALTER PROGRAM command to change the attribute values of a PROGRAM object description that was previously added to the PATHMON configuration file. For a list of the PROGRAM object attributes, see the [SET PROGRAM Command](#) on page 11-15.

```
ALTER [ PROGRAM ] program-name

    { , program-attribute
      { , RESET program-keyword
        { , RESET (program-keyword [ , program-keyword ]... ) }
      }

program-keyword is:

    ERROR-ABORT
    OWNER
    PRINTER
    SECURITY
    TCP
    TMF
    TYPE program-type
```

PROGRAM *program-name*

specifies the name of a previously defined and added PROGRAM object.

program-attribute

specifies an attribute describing a PROGRAM object. A PROGRAM object attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET PROGRAM command. Use any of the attributes listed for the SET PROGRAM command.

RESET

changes an existing attribute of a PROGRAM object to the default value.

If an attribute is not required and no default value exists, this option deletes the value set for the attribute.

If an attribute is required and no default value exists, the RESET option generates a syntax error and leaves the current attribute value unchanged.

program-keyword

specifies a single SET PROGRAM command attribute keyword or several keywords separated by commas and enclosed in parentheses.

Consideration

If you set multiple values for the RUN PROGRAM TYPE command, the RESET option deletes only the type you specify. For example, if you define TYPE 2, TYPE 4, and TYPE 5 for a particular PROGRAM object and enter RESET TYPE 5, only TYPE 5 is deleted from the list of types for that PROGRAM object definition. If the type specified with the RESET option is the only type definition for the PROGRAM object, a syntax error is generated and the attribute is not changed.

Errors

The following table lists the most common error that can occur during the processing of the ALTER PROGRAM command:

This Message...	Is Displayed When...
1108 TOO MANY PROGRAM ENTRIES	<p>You have tried to alter a PROGRAM object but you have already added the maximum number of PROGRAM objects (that is, 4095) supported by the PATHMON process.</p> <p>The command failed because the PATHMON process executes the ALTER PROGRAM command by creating a new PROGRAM object with the altered values before deleting the existing PROGRAM object. When the maximum number of PROGRAM objects already exists, the PATHMON process does not have the space to create a new one.</p> <p>To avoid this problem, define and add the maximum number of PROGRAM objects allowed minus 1 (that is, 4094 PROGRAM objects).</p>

Examples

The following commands change the PROGRAM object attributes shown:

```
ALTER PROGRAM PROG-1, TCP TCP-1
ALTER PROG-2, RESET ERROR-ABORT
```

The following commands change multiple PROGRAM object attributes:

```
ALTER PROGRAM PROG-3, RESET (OWNER, SECURITY, TMF)
ALTER PROGRAM PROG-3, TYPE 5 (INITIAL CONV-EXP) &
, TYPE 4 (INITIAL LOGON)
```

The following command deletes a specific TYPE attribute from the PROGRAM object definition:

```
ALTER PROGRAM PROG-3, RESET TYPE 5
```

DELETE PROGRAM Command

Use the DELETE PROGRAM command to remove a PROGRAM object description from the PATHMON configuration file.

```
DELETE [ PROGRAM ] { program-name  
                     { ( program-name [ , program-name ] ... ) } }
```

program-name

specifies the name of a PROGRAM object. Use either a single name or several names separated by commas and enclosed in parentheses.

Consideration

The DELETE PROGRAM command deletes only the specified template used for creating temporary TERM objects—it does not delete the temporary TERM objects currently running and generated from that template.

To delete a temporary TERM object, you must first use the ABORT TERM or STOP TERM command, then use the DELETE TERM command. If the ERROR-ABORT attribute for the template is set to ON, the TCP automatically aborts a temporary TERM object if a program run error occurs.

Examples

The following commands delete the PROGRAM objects shown:

```
DELETE PROGRAM PROG-1  
DELETE (PROG-3, PROG-4, PROG-5)
```

INFO PROGRAM Command

Use the INFO PROGRAM command to display the attribute values defined in the PATHMON configuration file for a PROGRAM object.

```
INFO [ / OUT list-file / ]

{ [ PROGRAM ] program-name
  [ PROGRAM ] ( program-name [ , program-name ]... )
  PROGRAM *
}

[ , OBEYFORM ]
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

PROGRAM *program-name*

specifies the name of a PROGRAM object. You can specify either a single name or several names separated by commas and enclosed in parentheses.

PROGRAM *

displays the attribute values for all PROGRAM objects described in the PATHMON configuration file.

OBEYFORM

displays the information in the format you would use to set up a PATHMON configuration file; each attribute is displayed as a syntactically correct PATHCOM SET command. PATHCOM adds a RESET PROGRAM command before and an ADD PROGRAM command after each PROGRAM object description.

Considerations

- INFO PROGRAM displays all of the SET PROGRAM attribute values that you explicitly defined for the PROGRAM object. This command returns information about a PROGRAM object only after the description has been added to the PATHMON configuration file.
- OBEYFORM is an INFO command option for the PATHMON, PATHWAY, TCP, TERM, SERVER, and PROGRAM objects. By using the OUT *list-file* option and issuing an INFO command for each object in succession, you can build a new cold start command file that reflects any modifications you have made to your PATHMON environment. You must add the START PATHWAY command at the end of the OUT file before using the file to start a PATHMON environment.

Examples

The following command requests PROGRAM object attribute information for two specific PROGRAM objects:

```
INFO (PROG-3, PROG-5)
```

The following command:

```
INFO PROGRAM PROG-1, OBEYFORM
```

causes the PATHMON process to display the PROGRAM object attributes in the following format:

```
RESET PROGRAM
SET PROGRAM ERROR-ABORT ON
SET PROGRAM OWNER \*.8,8
SET PROGRAM SECURITY "G"
SET PROGRAM TCP TCPA
SET PROGRAM TMF ON
SET PROGRAM TYPE T16-6530 (INITIAL MENU)
ADD PROGRAM PROG-1
```

The following command directs the information to a particular file:

```
INFO/OUT PROGFILE/PROGRAM *, OBEYFORM
```

The following sequence of commands captures a configuration for a complete PATHMON environment (minus the START PATHWAY command) in the command file called NEWCONFIG:

```
=INFO/OUT NEWCONFIG/PATHMON, OBEYFORM
=INFO/OUT NEWCONFIG/PATHWAY, OBEYFORM
=INFO/OUT NEWCONFIG/TCP *, OBEYFORM
=INFO/OUT NEWCONFIG/TERM *, OBEYFORM
=INFO/OUT NEWCONFIG/SERVER *, OBEYFORM
=INFO/OUT NEWCONFIG/PROGRAM *, OBEYFORM
```

Note. For information about the PATHMON, PATHWAY, and SERVER objects listed in this example, see the *TS/MP System Management Manual*.

RESET PROGRAM Command

Use the RESET PROGRAM command to change the values for the PROGRAM object attributes (listed under *program-keyword* in the syntax) from the values that you defined with the SET PROGRAM command to the defaults. This command does not change the attributes of a PROGRAM object already added to the PATHMON configuration file.

```
RESET PROGRAM [ program-keyword [ , program-keyword ]... ]
```

program-keyword is:

```
ERROR-ABORT  
OWNER  
PRINTER  
SECURITY  
TCP  
TMF  
TYPE program-type
```

program-keyword

specifies one or more attributes to be reset.

If you omit *tcp-keyword*, this command resets all attribute values to the defaults.

Considerations

- If you invoke the RESET PROGRAM command after the SET PROGRAM command but before the ADD PROGRAM command, the attribute values you specify revert to the default values.
- Some required PROGRAM object attributes have no default values. If you include a required attribute in the RESET PROGRAM command, the attribute is set to a null value and must be set again before you add the PROGRAM object description to the PATHMON configuration file.

If an attribute is not required and does not have a default value, the RESET PROGRAM command deletes the value for the attribute.

- If you set multiple values for an attribute (for example, the TYPE attribute), the RESET PROGRAM command deletes all the values set for the attribute.

Examples

The following command resets all attribute values:

```
RESET PROGRAM
```

The following commands reset the values for the specified attributes:

```
RESET PROGRAM TMF  
RESET PROGRAM TCP, OWNER  
RESET PROGRAM TYPE 3, TYPE 4
```

RUN PROGRAM Command

Use the RUN PROGRAM command to initiate the execution of a SCREEN COBOL program on a device interoperating with your PATHMON environment. When this command is issued, the PATHMON process performs the following:

1. Creates a temporary TERM object based on the configuration information in the PROGRAM object template. (This is similar to the ADD TERM command.) Assigns an object name beginning with a number to distinguish it from configured TERM objects.
2. If necessary, starts the TCP that controls the device.
3. Starts the newly created temporary TERM object. (This is similar to the START TERM command.)
4. Passes control of the device to the TCP while the TCP executes the SCREEN COBOL program.
5. After the screen program terminates and if no other programs are using the same temporary TERM object, or when the TERM object is aborted, deletes the temporary TERM object. (This is similar to the DELETE TERM command.)

If you want PATHCOM to be available before the SCREEN COBOL program completes, specify the NOWAIT parameter.

The RUN PROGRAM command must be issued interactively. This command fails if PATHCOM encounters the command in a command file.

```

RUN [ PROGRAM ] program-name
    [ , program-parameter [ , program-parameter ]... ]

program-parameter is:
    PRINTER { file-name [ IS-ATTACHED ] }
             { IS-ATTACHED }
    FILE file-name
    TYPE program-type
    NOWAIT
  
```

program-name

specifies the name of a previously defined PROGRAM object.

program-parameter

specifies one or more parameters to be run.

PRINTER

specifies the printer to be used for the SCREEN COBOL PRINT SCREEN verb.

An attached printer either physically plugs into the terminal (for example, a 6520 or 6530 terminal) or connects to the same line controller (for example, an IBM 3270 compatible terminal using AM3270) as the terminal on which the SCREEN COBOL program runs.

file-name

specifies the file name for the print device. You usually use *file-name* when the printer is not attached or when multiple attached printers are connected to a line controller. This parameter overrides the current value configured for the PRINTER parameter.

The specified file name is placed in the SCREEN COBOL special register TERMINAL-PRINTER when the program starts running on a terminal. While a screen program runs, it can modify the file name in its TERMINAL-PRINTER register and then use the new file name.

If you omit *file-name*, the SCREEN COBOL TERMINAL-PRINTER special register is set to SPACES (that is, ASCII blanks).

If a printer is not attached, the SCREEN COBOL verb PRINT-SCREEN obtains the printer's file name from the special register TERMINAL-PRINTER.

IS-ATTACHED

directs the copy of the screen image to a printer that is either physically attached to the terminal or on the same line controller as the terminal on which the program runs. IS-ATTACHED can be used with the following:

T16-6530 Directs copy to a printer attached directly to the terminal.

T16-6520 Directs copy to a printer attached directly to the terminal.

IBM-3270 Directs copy to the device that is attached to the same control unit as the terminal.

If you specify PRINTER IS-ATTACHED for a 6520, 6530, or 6540 terminal, the SCREEN COBOL special register TERMINAL-PRINTER is ignored.

FILE *file-name*

specifies the name of a terminal or a process emulating a terminal that is used as the I/O device for this PROGRAM object. This parameter allows an application to write a front-end process that emulates a standard terminal type and acts as a link between the PATHMON environment and a nonstandard terminal or an intelligent device.

If you omit this parameter, the SCREEN COBOL program runs on the device where the PATHCOM process is running. Control is returned to the PATHCOM process only when the SCREEN COBOL program terminates.

The *file-name* parameter must be represented in network form. It must begin with a \$ (dollar sign) followed by one to six alphanumeric characters, and the first character must be alphabetic. The *file-name* cannot exceed a total of seven characters including the \$ symbol.

Specifying a terminal name as the file allows you to run PATHCOM on one terminal and the SCREEN COBOL program on another terminal. However, unless NOWAIT is specified, you cannot use the terminal running PATHCOM, because this terminal has to wait for the SCREEN COBOL program to finish running on the other terminal.

file-name is required if NOWAIT is specified and must be a name that is different from the file name of the PATHCOM terminal.

For example, if the following command is issued from \$TERMA:

```
=RUN PROGRAM SALES, FILE $TERMB
```

the PATHCOM process running on \$TERMA must wait for the program SALES to finish running on \$TERMB.

TYPE *program-type*

specifies the type of device on which the screen program runs. The possible values are:

1: <i>n</i>	or	IBM-3270: <i>n</i>	IBM 3270 terminal
3:0	or	T16-6520:0	6520 terminal
4:0	or	T16-6530:0	6530 terminal
5:0	or	CONVERSATIONAL	Conversational-mode terminal
6:0	or	T16-6540:0	6540 terminal emulating a 6530 terminal
7: <i>n</i>	or	INTELLIGENT: <i>n</i>	Intelligent device
8:0	or	T16-6530WP:0	6530WP terminal

A PROGRAM object definition can have more than one TYPE parameter value associated with it. For instance, in the following example a value is specified for an IBM 3270 terminal, a 6530 terminal, and an intelligent device:

```
SET PROGRAM TCP TCP1
SET PROGRAM TYPE IBM-3270 (INITIAL SC01-IBM)
SET PROGRAM TYPE T16-6530 (INITIAL SC01-T16)
SET PROGRAM TYPE INTELLIGENT (INITIAL SC01-INT)
ADD PROGRAM PROG1
```

If you do not specify a TYPE parameter value, the PATHMON process issues a file-system DEVICEINFO query to the device the temporary TERM will run on. The reply from the DEVICEINFO query is used to determine the appropriate device type. If the device type derived from the DEVICEINFO reply is not defined in the PROGRAM object definition, the RUN PROGRAM command fails with an error 1086.

NOWAIT

causes the program to execute concurrently with PATHCOM, so that the PATHCOM prompt returns immediately to your screen. To use the NOWAIT option, you must specify a terminal that is different from the terminal on which you give the RUN PROGRAM command.

If you do not specify this option, the PATHCOM prompt does not return to your screen until after the program has completed running.

Considerations

- If a RUN PROGRAM command is issued from a 6530 terminal and no PROGRAM template is defined for that terminal type, but a PROGRAM object is defined for a 6520 terminal, the TCP executes the PROGRAM object as if it were defined for a 6520 terminal. The features that are unique to the 6530 terminal, however, do not function.
- The PATHMON process supports subtype 30 processes that function as devices managed by PROGRAM objects. Subtype 30 processes are nonprivileged processes that emulate terminals and communication devices. Because the PATHMON process determines the device type by directly polling the emulator process, you do not have to specify a value for the RUN PROGRAM TYPE parameter.

Errors

The following table lists the most common error that can occur during the processing of the RUN PROGRAM command:

This Message...	Is Displayed When...
ERROR *1148* MAXIMUM NUMBER OF WAITED RUN PROGRAM REQUESTS EXCEEDED	You attempted to initiate a waited RUN PROGRAM request, but the PATHCOM limit of 100 concurrent waited requests has already been reached. The RUN PROGRAM command fails.
1086 TERM TYPE NOT DEFINED FOR PROGRAM	You specified a TYPE parameter value but the device type you entered is not defined in the PROGRAM definition. The RUN PROGRAM command fails.

Examples

The following command starts the PROGRAM PROG-1:

```
RUN PROGRAM PROG-1
```

The following command directs a copy of the screen image to the printer that is attached to the terminal:

```
RUN PROGRAM PROG-2, PRINTER IS-ATTACHED
```

The following command directs a copy of the screen image to an unattached printer:

```
RUN PROG-3, PRINTER $LP
```

The following command directs the SALES program to run under control of process \$FE.#TERMA, which emulates a 6530 terminal:

```
RUN SALES, FILE $FE.#TERMA, TYPE T16-6530
```

The following command allows the SALES program to run from an intelligent device using the WRITEREAD protocol:

```
RUN SALES, TYPE INTELLIGENT
```

SET PROGRAM Command

Use the SET PROGRAM command to establish the values for the PROGRAM object attributes.

```
SET PROGRAM program-attribute [ , program-attribute ]...
```

program-attribute is:

```
TCP tcp-name
TYPE program-type (      INITIAL program-unit-name
                        [ , BREAK { ON | OFF }      ]
                        [ , DISPLAY-PAGES number    ]
                        [ , ECHO { ON | OFF | CURRENT } ]
                        [ , EXCLUSIVE { ON | OFF }    ]
                        [ , IOPROTOCOL { 0 | 1 }      ]
                        [ , MAXINPUTMSGs number      ]
                        [ , TCLPROG file-name        ]
                        [ , TRAILINGBLANKS { ON | OFF } )... ]

ERROR-ABORT { ON | OFF }
LIKE program-name
OWNER owner-id
PRINTER { file-name [ IS-ATTACHED ] }
        { IS-ATTACHED }
SECURITY security-attribute
TMF { ON | OFF }
```

program-attribute

The TCP and TYPE attributes have no defaults; you must specify them before adding a PROGRAM object definition. The rest of the attributes have default values.

For information about configuration limits, see [Appendix C, Configuration Limits and Defaults](#).

TCP *tcp-name*

specifies the name of the TCP that interprets the SCREEN COBOL program. If the TCP is not running when you issue the RUN PROGRAM command, the PATHMON process starts it.

This attribute is required.

TYPE *program-type*

specifies the type of program, the name of the screen program that the TCP uses when starting the terminal, and specific terminal and program options. The possible values are:

1 or IBM-3270	IBM 3270 terminal
3 or T16-6520	6520 terminal
4 or T16-6530	6530 terminal
5 or CONVERSATIONAL	Conversational-mode terminal
6 or T16-6540	6540 terminal emulating a 6530 terminal
7 or INTELLIGENT	Intelligent device
8 or T16-6530WP	6530WP terminal

For more information on specifying device types for PROGRAM objects, see the TYPE attribute of the [RUN PROGRAM Command](#) on page 11-10.

This attribute is required.

INITIAL *program-unit-name*

specifies the name of the SCREEN COBOL program unit that the TCP executes when starting the application.

This attribute is required.

BREAK { ON | OFF }

specifies whether the TCP accepts the Break key function for its terminals running in conversational mode.

If a process is running as a terminal emulator, the value for this attribute always defaults to OFF.

If you omit this attribute, the default is OFF. This attribute is effective only for terminals running in conversational mode.

ON

directs the TCP to accept the Break key function. If the TCP is executing a SCREEN COBOL program in conversational mode, the terminal operator can press the Break key to prematurely terminate ACCEPT and DISPLAY statements. If the terminal operator presses the Break key while the TCP is executing a DISPLAY statement, the terminal output stops and the DISPLAY statement is terminated normally with the special register TERMINATION-SUBSTATUS set to 1. If the Break key is not used, TERMINATION-SUBSTATUS is set to 0 after the completion of a DISPLAY statement.

If the terminal operator presses the Break key while the TCP is executing an ACCEPT statement that is coded with an ESCAPE ON ABORT clause, the TCP stops accepting terminal input and terminates the ACCEPT statement without changes to the working storage section. Pressing the Break key has no effect on the TCP when the ACCEPT statement is coded without an ESCAPE ON ABORT clause. If the TCP is not executing either of the preceding statements, pressing the Break key has no effect on the TCP process.

OFF

directs the TCP to ignore the Break key function.

DISPLAY-PAGES *number*

specifies the depth of the terminal's screen caching. DISPLAY-PAGES determines the maximum number of screen displays that are stored in the terminal's memory. If you omit this attribute, the default is used. If you specify a value greater than the defined maximum, the defined maximum is used.

Terminal Type	Default Value	Defined Minimum	Defined Maximum
T16-6540	7	1	16*
T16-6530WP	7	1	8*
T16-6530	7	1**	8*
T16-6520	3	1**	3
IBM-3270	1	1	1
TS530	8	1	8*

* This is the absolute maximum allowed for the TCP. However, if the terminal or terminal emulator has less memory, the working value is negotiated downward.

** Although the PATHMON process allows you to set the number of pages to 1, some terminals or terminal emulators do not honor that setting. For example, the T16-6520 terminal always sets its number of display pages to 3, regardless of the user setting.

You select the appropriate DISPLAY-PAGES value based on the following application-dependent information:

- The maximum number of fields per screen that is declared in the SCREEN COBOL program
- The average number of fields per screen the terminal type can handle for the particular setting of the DISPLAY-PAGES option

Terminal memory is divided into two separate areas: the screen image area and the field attribute area. The DISPLAY-PAGES attribute determines how much terminal memory is used for screen images as opposed to how much is used for field attributes. The more memory assigned to one, the less is available to the other. Therefore, if your application has many fields per screen, you should set DISPLAY-PAGES to a small value to leave room for the field

attributes. If you specify too many display pages, you might receive the terminal error ALL FIELDS USED.

For detailed information on how to select the appropriate value for the DISPLAY-PAGES attribute, see [Appendix F, Setting the DISPLAY-PAGES Parameter](#).

ECHO { ON | OFF | CURRENT }

overrides the terminal echo mode (that is, the mode configured for a terminal at system generation time) so that character input appears correctly on the terminal. The ECHO attribute supports terminals running in conversational or intelligent mode.

If you omit this attribute, the default is ON.

ON

directs input to appear on the terminal screen.

OFF

corrects the display by suppressing the duplicate characters (if a terminal is configured so that it displays input characters). If you define this option for the supported terminal types T16-6520, T16-6530, and T16-6540, no input characters appear on the terminal screen.

CURRENT

directs the TCP to obtain the operating mode of the terminal from the operating system and to use that mode when it opens the terminal. If you specify this option for a process that is running as a terminal emulator, this attribute value defaults to ON.

EXCLUSIVE { ON | OFF }

specifies whether a TCP should open a terminal for exclusive or shared access.

If you omit this attribute, the default is OFF.

ON

directs a TCP to open the terminal for exclusive access. The TCP cannot allow simultaneous access of any kind to the terminal by any other process.

OFF

directs a TCP to open the terminal for shared access. The TCP allows simultaneous read or write access to the terminal by other processes.

IOPROTOCOL { 0 | 1 }

specifies the protocol the TCP follows when a SCREEN COBOL SEND MESSAGE statement completes with a timeout or an unsolicited message.

If you omit this attribute, the default is 0.

0

directs a TCP to issue a CANCEL procedure to cancel a SEND MESSAGE I/O. The I/O is cancelled because an unsolicited message arrives or a SEND MESSAGE TIMEOUT occurs.

1

directs a TCP to use a CONTROL-26 protocol to cancel a SEND MESSAGE I/O. The I/O is cancelled because an unsolicited message arrives or a SEND MESSAGE TIMEOUT occurs.

If IOPROTOCOL 1 is specified, the front-end process must be written according to rules described in the *TS/MP Pathsend and Server Programming Manual*. Front-end processes can use this attribute to avoid loss of data.

MAXINPUTMSG *number*

specifies the maximum number of unsolicited messages that the TCP queues for the requester associated with the PROGRAM object. If this attribute is exceeded, the TCP returns a TERM^QUEUE^FULL error to the originator of the unsolicited message.

number must be between 0 and 2045. If you omit this attribute, the default is 0.

TCLPROG *file-name*

specifies the name of the SCREEN COBOL object library file that contains the program units. The TCP checks this file name for the requested program unit before checking the SET TCP TCLPROG file name. If you omit this option, the TCP uses the file defined for the SET TCP TCLPROG attribute.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the *NonStop* TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

In a production environment it is better not to specify a TCLPROG filename for PROGRAM (and configured TERM) objects. If you specify a TCLPROG filename, the TCP checks the file every time a SCREEN COBOL program unit is requested.

TRAILINGBLANKS { ON | OFF }

specifies whether the TCP suppresses the display of trailing ASCII blank characters in output lines sent to terminals by the SCREEN COBOL DISPLAY and SEND MESSAGE verbs.

If you omit this attribute, the default is ON.

ON

directs the TCP not to suppress trailing ASCII blanks.

OFF

directs the TCP to suppress trailing ASCII blanks.

This attribute affects terminals defined as TYPE CONVERSATIONAL and affects programs defined as TYPE INTELLIGENT that use delimited format messages. This attribute has no effect on VARYING1 and VARYING2 message formats.

ERROR-ABORT { ON | OFF }

specifies whether the TCP aborts or suspends the terminal if an application error occurs.

If you omit this attribute, the default is ON.

ON

directs the TCP to abort the terminal.

OFF

directs the TCP to suspend the terminal.

LIKE *program-name*

sets the attribute values for a PROGRAM object to those of a previously added PROGRAM object.

OWNER *owner-id*

specifies the user ID authorized to issue the RUN PROGRAM command. The user ID must be known to the system on which PATHCOM is running. Use this attribute in conjunction with the SECURITY attribute.

The value of *owner-id* is one of the following:

```
[ \system-number.] group-number, user-number
[ \node.] group-name.user-name
```

If the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command is set to ON and you omit the node name, or if you specify * for the *ownerid* node name or system number, the node name used is the node on which the PATHMON process is currently running. The *group-name* and *user-name* are configured as you specify.

If the NODEINDEPENDENT attribute of the SET PATHWAY command is set to OFF and you omit the node name (or system number) parameter, the node value defaults to the current default node of the user who started PATHCOM, and the *group-name* and *user-name* are configured as you specify.

If you omit this attribute, the node value defaults to the current default node of the user who started PATHCOM, and the *group-name* and *user-name* values default to the user ID of the user who started PATHCOM. (This attribute does not use the defaults established by the CMDVOL command.)

PRINTER

specifies the printer to be used for the SCREEN COBOL PRINT SCREEN verb.

An attached printer either physically plugs into the terminal (for example, a 6520 or 6530 terminal) or connects to the same line controller (for example, an IBM 3270 compatible terminal using AM3270) as the terminal on which the SCREEN COBOL program runs.

file-name

specifies the file name for the print device. You usually use *file-name* when the printer is not attached or when multiple attached printers are connected to a line controller. This attribute overrides the current value configured for the PRINTER attribute.

If you do not specify a node name, the default node for file-name expansion can be affected by several factors, including values you specify for the NonStop TS/MP CMDVOL command and the NODEINDEPENDENT attribute of the TS/MP SET PATHWAY command. For details, see [Specifying Node Independence](#) on page 2-5.

The specified file name is placed in the SCREEN COBOL special register TERMINAL-PRINTER when the program starts running on a terminal. While a screen program runs, it can modify the file name in its TERMINAL-PRINTER register and then use the new file name.

If you omit *file-name*, the SCREEN COBOL TERMINAL-PRINTER special register is set to SPACES (that is, ASCII blanks).

If a printer is not attached, the SCREEN COBOL verb PRINT-SCREEN obtains the printer's file name from the special register TERMINAL-PRINTER.

IS-ATTACHED

directs the copy of the screen image to a printer that is either physically attached to the terminal or on the same line controller as the terminal on which the program runs. IS-ATTACHED is one of the following:

T16-6530 Directs copy to a printer attached directly to the terminal.

T16-6520 Directs copy to a printer attached directly to the terminal.

IBM-3270 Directs copy to the device that is attached to the same control unit as the terminal.

If you specify PRINTER IS-ATTACHED for a 6520, 6530, or 6540 terminal, the SCREEN COBOL special register TERMINAL-PRINTER is ignored.

SECURITY *security-attribute*

specifies the PATHCOM users who can issue a RUN PROGRAM command for this PROGRAM object. This attribute controls only who can execute a RUN PROGRAM command; it does not control who can execute a CALL statement from another SCREEN COBOL program unit.

The security attributes are the same as the operating system security attributes. The possible values are:

"A" Any local user

"G" A group member or owner

"O" Owner only

" - " Local super ID

"N" Any local or remote user

"C" Any member of owner's community (local or remote user having same group ID as owner)

"U" Any member of owner's user class (local or remote user having same group ID and user ID as owner)

Quotation marks are required. If you omit this attribute, the default is "N".

TMF { ON | OFF }

specifies whether the TMF subsystem runs with this PROGRAM object. This option is intended to be used only as a debugging aid. For more information about this attribute, see [Appendix E, Setting TMF Parameters](#).

If you omit this attribute, the default is ON.

- ON invokes TMF auditing when necessary. The ABORT-TRANSACTION, BEGIN-TRANSACTION, END-TRANSACTION, and RESTART-TRANSACTION verbs perform the functions described in the *Pathway SCREEN COBOL Reference Manual*.
- OFF prevents TMF auditing. The TRANSACTION-ID special register is set to SPACES after the BEGIN-TRANSACTION verb executes.

Considerations

- Before adding a PROGRAM description to the PATHMON configuration file, you must define values for all of the required attributes listed in this command.
- A TCP can execute the same SCREEN COBOL programs on different types of terminals so that you can run programs written for conversational-mode terminals on some block-mode terminals. If the SCREEN COBOL TERMINAL IS clause specifies CONVERSATIONAL and the PROGRAM TYPE attribute is T16-6530, the TCP executes the program on the 6530 terminal in conversational mode. If the TERMINAL IS clause is omitted and the TCP attempts to execute this same program, an error is returned.

Examples

The following commands define a subset of the PROGRAM object attributes:

```
SET PROGRAM TCP TCP-1
SET PROGRAM TYPE 4 (INITIAL MAINMENU)
SET PROGRAM OWNER 8,9
SET PROGRAM SECURITY "G"
```

The following command defines multiple terminal options:

```
SET PROGRAM TYPE 5 (INITIAL LOGON, BREAK ON, ECHO ON)
```

SHOW PROGRAM Command

Use the SHOW PROGRAM command to display a subset of the PROGRAM object attributes in alphabetic order. This command displays the attribute values for a PROGRAM object not yet added to the PATHMON configuration file.

```
SHOW [ / OUT list-file / ] PROGRAM
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

Considerations

- When PATHCOM first starts and after a RESET PROGRAM command executes, the SHOW PROGRAM command displays only the required PROGRAM object attributes and a subset of the optional attributes that have default values. A question mark appears for the required attribute values that you must set before adding the PROGRAM object.
- SHOW PROGRAM displays all of the attribute values that you explicitly set with the SET PROGRAM command.
- For attributes that include *Guardian* operating environment file names, SHOW PROGRAM displays the node-name part of the file name as either `\nodename` or `*`. The `\nodename` form is used if the node was explicitly specified by the user. The `*` form is used if the attribute was specified to default to the node where the PATHMON process is running.
- After an ADD PROGRAM command, the SHOW PROGRAM command continues to display the attribute values held in PATHCOM for the previously added PROGRAM object. Use the RESET PROGRAM command to change all of the values to their defaults, or use the SET PROGRAM command to replace specific attribute values.

Example

If you issue the SHOW PROGRAM command before setting the PROGRAM object attributes, the following information is displayed:

```
PROGRAM
ERROR-ABORT ON
OWNER \TS.8,8
SECURITY "N"
TCP ?
TMF ON
TYPE ?
```

The following command directs the SHOW PROGRAM display to the file named PROGFILE:

```
SHOW/OUT PROGFILE/PROGRAM
```

Wild Card Support for PROGRAM Commands

The INFO PROGRAM command in PATHCOM allows management and operation of all PROGRAM objects configured under a PATHMON environment by using the wild card "*". It can be applied to a subset of configured PROGRAM objects.

Note. The above mentioned wild card support is applicable only to TS/MP 2.3 PATHMON and PATHCOM. For more information on TS/MP 2.3, see the *TS/MP Release Supplement Manual*.

Example

The following command displays the PROGRAM object attribute values configured under the PATHMON environment whose name starts with EXPR:

```
INFO PROGRAM EXPR*
```

The system displays this output:

```
PROGRAM EXPROG
```

```
  ERROR-ABORT ON
```

```
  OWNER  \ACS.255,255
```

```
  SECURITY "N"
```

```
  TCP EX-TCP
```

```
  TMF OFF
```

```
  TYPE T16-6520 (INITIAL EXAMPLE)
```

```
PROGRAM EXPROG1
```

```
  ERROR-ABORT ON
```

```
  OWNER  \ACS.255,255
```

```
  SECURITY "N"
```

```
  TCP EX-TCP
```

```
  TMF OFF
```

```
  TYPE T16-6520 (INITIAL EXAMPLE)
```

```
PROGRAM EXPROG2
```

```
  ERROR-ABORT ON
```

```
  OWNER  \ACS.255,255
```

```
  SECURITY "N"
```

```
  TCP EX-TCP
```

```
  TMF OFF
```

```
  TYPE T16-6520 (INITIAL EXAMPLE)
```


12 Tell Message Commands

The commands for controlling tell messages are:

- DELETE TELL
- INFO TELL
- TELL TERM

These commands control messages that a TCP displays to the transaction process terminal operators. The commands can be directed to configured and temporary TERM objects.

The SCREEN COBOL TELL-ALLOWED special register controls the issuing of these tell messages. When the TELL-ALLOWED special register is set to YES, tell messages can be displayed for the specified terminal. When this register is set to NO, tell messages are postponed.

Tell messages are not supported for intelligent devices, except for personal computers and workstations connected to a NonStop system using the HP NonStop Remote Server Call/MP (RSC/MP) product.

DELETE TELL Command

Use the DELETE TELL command to delete a pending tell message.

`DELETE TELL number`

number

specifies the number of the pending tell message.

Examples

The following commands delete the specified pending tell messages:

```
DELETE TELL 4  
DELETE TELL 10
```

INFO TELL Command

Use the INFO TELL command to display the text of a pending tell message.

```
INFO [ / OUT list-file / ] TELL { number }  
                                   { * }
```

OUT *list-file*

directs output to the named list file; this could be a DEFINE name. If this option is omitted, the PATHMON process directs the output to the PATHCOM list file; this is typically the home terminal.

number

specifies an integer that identifies the message whose text is to be displayed.

*

displays the text of all pending tell messages.

Examples

The following commands display the text of the specified pending tell messages:

```
INFO TELL 46  
INFO / OUT TELFILE / TELL *  
INFO TELL *
```

TELL TERM Command

Use the TELL TERM command to display a message on a terminal or set of terminals. The TCP waits for the terminal operator to complete the current screen before displaying the message, so normal operation is not disrupted.

```
TELL { [ TERM ] termname-list
      { TERM * [ , option [ , option ] ] } , "string"
```

option is:

```
STATE state-type
TCP tcp-name
```

termname-list

specifies the name(s) of one or more terminals (either a configured TERM object or a temporary TERM object created with the RUN PROGRAM command) on which a message is to be displayed. If an INTELLIGENT terminal is included on the list, it is ignored.

"*string*"

specifies the message text. The quotation mark characters are required.

TERM *

displays a message on all terminals (configured or temporary TERM objects) controlled by all TCPs.

STATE *state-type*

displays a message on all terminals in the specified state. Possible values are:

RUNNING	NOT RUNNING
STOPPED	NOT STOPPED
SUSPENDED	NOT SUSPENDED

You can substitute SEL for STATE.

TCP *tcp-name*

displays a message on all TERM objects controlled by the specified TCP.

Considerations

- Each terminal has a circular queue for tell messages. If the queue is full when a new tell message is sent, the oldest tell message in the queue is overwritten by the new message. You specify the size of the queue using the MAXTELLQUEUE attribute in the SET PATHWAY command.
- When you issue a TELL command, the tell message is assigned a number between zero and the value of MAXTELLS. The PATHMON process deletes a tell message automatically when all destination terminals acknowledge receiving the message.
- To use the TELL TERM command, you must have a licensed copy of the HP NonStop Pathway/iTS product on your system; otherwise, the PATHMON process returns an error.

Examples

The following commands display the messages given on the named terminals:

```
TELL TERM-4,"Check with Accounting for new procedures"  
TELL TERM(TERM-5,TERM-6),"Sign-off is at 8:45 p.m."
```

The following command displays the specified message on all terminals:

```
TELL TERM *, "The system will be unavailable after 4 today."
```


TCP Messages (Numbers 3000-3999)

This section lists the messages for errors or other conditions reported by terminal control processes (TCPs).

General Information

All messages returned by the PATHMON environment are logged by the PATHMON process; however, messages in some number ranges represent errors reported to the PATHMON process by other processes. The message number identifies the process that reported the error, as follows:

Message Numbers in This Range...	Represent Errors Reported by...
1000 through 1999	The PATHMON process
2000 through 2999	The PATHCOM interface
3000 through 3999	Link managers (TCPs or LINKMON processes)

Note. This manual describes only messages generated by TCPs (messages 3000 through 3999). A few of these messages can also be reported by LINKMON processes; such messages are described also in the “Link Manager Messages” section of the *TS/MP System Management Manual*. Messages generated by the PATHMON process and the PATHCOM interface (messages 1000 through 1999 and 2000 through 2999, respectively), are described in the *TS/MP System Management Manual*. Bear in mind that some messages pertaining to HP NonStop Pathway/iTS objects, such as TCP, TERM, and PROGRAM objects, are generated by the PATHMON process or the PATHCOM interface rather than by TCPs. For information about those messages, see the *TS/MP System Management Manual*. In addition, log messages generated by LINKMON processes are described in the *TS/MP System Management Manual*.

Additional Error Information

Some messages pertaining to terminal errors include additional information on the reason for the terminal error. The additional information includes the following fields returned from the STATUS TERM, DETAIL command. The additional information is placed at the end of the error-message text and starts with three dashes (---):

Additional Error Information Field	Description
INST CODE <i>instruct</i>	The instruction code of the program unit.
AT OFFSET <i>%offset</i>	The instruction address of the program unit.
IN PROGRAM UNIT <i>unit</i> (<i>version</i>)	The name of the program unit followed by the version of the program unit in parentheses.
IN TCLPROG <i>tclprog</i>	The file where the program unit is located.

For example:

```
ERROR - *3161* TERM TERM-1, I/O ERROR --- INST CODE: SEND
        AT OFFSET %37 IN PROGRAM UNIT PROG-X(1) IN TCLPROG
        \SYS.$VOL.SUBVOL.POBJ
```

Note. The additional information on terminal errors, which is appended to the standard error message text, significantly increases the size of messages sent to a log file.

The following TCP errors include additional information:

- Any error in the 3000-3999 range that applies to a particular terminal
- Any error message in the 3160 through 3196 range, except errors 3165 and 3175

Operating System Error Numbers

Some PATHMON environment messages include an operating system error number. Although the error number is typically identified as a file-system error, it might also represent a sequential I/O (SIO) error or other specialized error.

SCREEN COBOL Errors

When an error originates in the SCREEN COBOL program, use the STATUS TERM, DETAIL command to obtain the name, version, and instruction address of the program unit being executed. You can isolate the problem to a paragraph in the source program using the information obtained from the STATUS TERM, DETAIL command and the compiler listing with a MAP.

TCP Messages

The following messages are generated by TCPs; however, LINKMON processes may also be the subject of some of the messages:

3001

<pre>*3001* TERM <i>term-name</i>, INVALID PSEUDOCODE DETECTED (%<i>p-reg</i>)</pre>
--

Cause. A SCREEN COBOL object file (COD) is corrupted, or the SCREEN COBOL compiler generated invalid code. This is an internal error.

Effect. The terminal is suspended.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version

- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the Global Customer Support Center (GCSC), supply your system number and the numbers and versions of all related products as well.

3002

3002 TERM *term-name*, DEPENDING VARIABLE VALUE TOO BIG

Cause. A DEPENDING ON data item was larger than the limit specified in the OCCURS clause.

Effect. The terminal is aborted; the operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3003

3003 TERM *term-name*, INVALID SUBSCRIPT VALUE

Cause. The value of a subscript expression is either 0 or larger than the limit implied by the OCCURS clause.

Effect. The TERM is aborted; the operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3004

3004 TERM *term-name*, SCREEN RECOVERY EXECUTED ILLEGAL INSTRUCTION

Cause. The TCP detected a SCREEN COBOL verb that is not allowed during screen recovery.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3005

3005 TERM *term-name*, CALL: ACTUAL NUMBER OF PARAMETERS
MISMATCHES FORMAL

Cause. The number of parameters passed in the USING clause of a CALL statement does not equal the number of parameters given in the Procedure Division header of the called program.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3006

3006 TERM *term-name*, CALL: ACTUAL PARAMETER SIZE MISMATCHES
FORMAL

Cause. The maximum size of a parameter passed in the USING clause of a CALL statement does not equal the maximum size of the corresponding parameter given in the Procedure Division header of the called program.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3007

3007 TERM *term-name*, SCREEN OPERATION DONE WITHOUT BASE
DISPLAYED

Cause. A screen-manipulation operation was attempted before a DISPLAY BASE statement was executed.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3008

```
*3008* TERM term-name, ILLEGAL DATA REFERENCE
```

Cause. There was an attempt to reference data outside the SCREEN COBOL program's local data area.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3009

```
*3009* TERM term-name, REFERENCED SCREEN IS ILLEGAL FOR  
TERMINAL TYPE (%p-register)
```

Cause. An attempt was made to execute a SCREEN COBOL program unit on an incompatible device. Or, if the device type is compatible, then a *NonStop* TS/MP internal error has occurred.

Effect. The operation fails.

Recovery. Check that the terminal type that was specified when the program unit was compiled is compatible with the physical terminal. If necessary, resolve the incompatibility. Otherwise, an internal error has occurred; contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3010

```
*3010* TERM term-name, INTERNAL ERROR IN TERMINAL FORMAT  
ROUTINES (%p-register)
```

Cause. There is an error in the terminal's format routines. This is an internal error.

Effect. The operation fails.

Recovery. Check that the terminal type that was specified when the program unit was compiled is compatible with the physical terminal. If necessary, resolve the incompatibility. Otherwise, an internal error has occurred; contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3011

```
*3011* TERM term-name, ILLEGAL TERMINAL TYPE SPECIFIED
(%p-register)
```

Cause. An attempt was made to execute a SCREEN COBOL program unit on an incompatible device. Or, if the device type is compatible, then an internal error has occurred.

Effect. The operation fails.

Recovery. Check that the terminal type that was specified when the program unit was compiled is compatible with the physical terminal. If necessary, resolve the incompatibility. Otherwise, an internal error has occurred; contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3012

3012 TERM *term-name*, SCREEN REFERENCED BUT NOT DISPLAYED
(*screen-number*)

Cause. A screen was referenced but is not displayed. The screen is identified by a screen number; 0 corresponds to the first screen declared in the program source.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3013

3013 TERM *term-name*, OVERLAY SCREEN DISPLAYED IN TWO AREAS

Cause. There was an attempt to display an overlay screen in more than one overlay area at the same time.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3014

3014 TERM *term-name*, ILLEGAL TERMINAL IO PROTOCOL WORD

Cause. The terminal control process (TCP) detected an invalid I/O protocol value or an internal TS/MP or Pathway/iTS error occurred.

Effect. The operation fails.

Recovery. If an invalid I/O protocol value was detected, contact the application programmer.

If an internal error occurred, contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3015

```
*3015* TERM term-name, ARITHMETIC OVERFLOW
```

Cause. A data-related error occurred during numeric manipulation. The error can be caused by an overflow, by dividing by zero, by an invalid value in a numeric data item (for example, blank spaces in a USAGE DISPLAY item), or by attempting to move a value into a data item that is not large enough to hold it.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3016

```
*3016* TERM term-name, TERMINAL STACK SPACE OVERFLOW (bytes)
```

bytes

indicates the number of bytes that would be needed to execute beyond the point of the error.

Cause. The limit configured for the MAXTERMDATA attribute in the SET TCP command was exceeded.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3017

```
*3017* TERM term-name, ERROR DURING TERMINAL OPEN (errnum)
```

Cause. A file-system error occurred while a TCP was opening the specified terminal.

Effect. The program terminates abnormally.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3018

```
*3018* TERM term-name, ERROR DURING TERMINAL IO (errnum)
```

Cause. A file-system error occurred while a TCP was performing a terminal I/O operation.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3019

```
*3019* TERM term-name, WRONG TRANSFER COUNT IN TERMINAL IO
```

Cause. The write count provided by the terminal control process (TCP) did not match the count written (returned by the terminal). This might indicate a terminal hardware problem.

Effect. The I/O operation fails.

Recovery. The INFO field returned by the STATUS TERM command displays the transfer count requested by the TCP. Run terminal diagnostics for the device and make any necessary repairs.

If this error is not caused by a hardware problem and a front-end process is being used, check that it is correctly coded to emulate the type of device that was specified. If a front-end process is not being used, locate the problem within the program.

3020

```
*3020* TERM term-name, CALLED PROGRAM UNIT NOT FOUND
```

Cause. The program unit specified in a CALL statement was not found in the TCLPROG files.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3021

```
*3021* TERM term-name, TRANSACTION MESSAGE SEND FAILURE
```

Cause. A SCREEN COBOL SEND operation to a server failed for the reason indicated in an earlier operator message.

Effect. If the transaction message was sent to the server, the system does not know how much processing was done by the server before the failure.

Recovery. Recovery depends on why the SEND operation failed as indicated by an earlier operator message. For additional information, see the description of that message.

3022

```
*3022* TERM term-name, SEND: SERVER CLASS NAME INVALID
```

Cause. The value specified in a SCREEN COBOL SEND statement does not have the format of a valid server class name.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3023

```
*3023* TERM term-name, PSEUDOCODE SIZE TOO BIG
```

Cause. The size of a program unit's code is greater than that allowed by the CODEAREALEN attribute.

Effect. The program terminates abnormally.

Recovery. Increase the value for the CODEAREALEN parameter. Use the compiler listing for the program unit to determine the minimum size that can be specified for the parameter.

3024

```
*3024* TERM term-name, TCLPROG DIRECTORY ENTRY IS BAD
```

Cause. The TCLPROG directory (DIR) file is corrupt. This is an internal error.

Effect. The program terminates abnormally.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version

- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3025

```
*3025* TERM term-name, TERMINAL INPUT DATA STREAM INVALID
```

Cause. Input data received from the terminal does not have the structure expected by the terminal control process (TCP), even after screen recovery and retries.

Effect. The I/O operation fails.

Recovery. Run diagnostics for the device. If a front-end process is being used, check that it is correctly coded to emulate the type of the device that was specified.

3027

```
*3027* TERM term-name, TRANSACTION MODE VIOLATION
```

Cause. The requested operation does not match the transaction mode state.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3028

```
*3028* TERM term-name, TRANSACTION I/O ERROR (errnum)
```

Cause. A file-system error occurred during a call to a TMF procedure.

Effect. The call to the TMF procedure fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3029

```
*3029* TERM term-name, TRANSACTION RESTART LIMIT REACHED
```

Cause. The TCP attempted to restart a transaction more than the number of times defined for the MAXTMFRESTARTS attribute.

Effect. The transaction is not restarted.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3030

```
*3030* TERM term-name, TMF NOT CONFIGURED
```

Cause. An attempt to use the TMF failed because TMF auditing is not configured in this system.

Effect. None.

Recovery. Configure TMF auditing or isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3031

```
*3031* TERM term-name, TMF NOT RUNNING
```

Cause. An attempt to use the TMF failed because the TMF software was not running.

Effect. None.

Recovery. Configure TMF auditing or isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3032

```
*3032* TCP tcp-name, TMF TFILE OPEN FAILURE (errnum)
```

Cause. File-system error occurred while the terminal control process (TCP) backup process was opening the TMF TFILE.

Effect. The open operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3036

```
*3036* TERM term-name, CANNOT CALL PU WITH TERMINAL-ERRORS  
DECLARATIVE
```

Cause. A program unit was called with a USE FOR TERMINAL-ERRORS statement from a USE FOR TERMINAL-ERRORS declarative.

Effect. The call fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3037

```
*3037* TERM term-name, ILLEGAL ACCEPT VARIABLE TIMEOUT VALUE
```

Cause. An invalid timeout value was used in an ACCEPT statement. The value was either less than 0 or too large to be represented in an INT(32) field.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3040

```
*3040* TERM term-name, INVALID NUMERIC ITEM - DESCRIPTOR  
NUMBER (number)
```

Cause. The field referenced in a verb does not contain numeric data. To find the item name that caused the error, find the indicated number in the DESC column and read across to the left.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1. The descriptor number appears on the compiler listing if you included the ?MAP or ?SMAP option in your program.

3041

3041 TERM *term-name*, INVALID PRINTER SPECIFICATION

Cause. An invalid printer was specified: either the printer for an IBM 3270 terminal is not attached to the same control unit as the terminal, or ATTACHED PRINT SCREEN was specified for a 6510 terminal.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3042

3042 TERM *term-name*, DEVICE REQUIRES INTERVENTION (*errnum*)

Cause. A file-system error occurred during a PRINT SCREEN operation. The error requires operator intervention.

Effect. The PRINT SCREEN operation terminates abnormally.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3043

3043 TERM *term-name*, PRINTER I/O ERROR (*errnum*)

Cause. A file-system error occurred during a PRINT SCREEN operation.

Effect. The PRINT SCREEN operation terminates abnormally.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3044

3044 TERM *term-name*, CALL: PROGRAM UNIT NAME INVALID

Cause. An invalid program unit name was specified in a CALL statement.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3050

```
*3050* TERM term-name, TERMINAL STOPPED BY PENDING REQUEST
```

Cause. The terminal stopped due to a pending stop request.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3051

```
*3051* TERM term-name, TERMINAL SUSPENDED BY PENDING REQUEST
```

Cause. The terminal was suspended because of a pending suspend request.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3052

```
*3052* TERM term-name, TERMINAL STOPPED BY PROGRAM
```

Cause. The terminal stopped because it encountered an exit in the highest-level program unit.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3053

```
*3053* TERM term-name, INVALID NUMERIC ITEM - INSTRUCTION  
ADDRESS (%address)
```

Cause. An operation that does not involve screen presentation, such as an ADD or MOVE operation, attempted to perform a numeric operation using nonnumeric data. The address is the octal address of the instruction that failed.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3054

```
*3054* TERM term-name, IO PROTOCOL DENIED (protocol-number)
```

Cause. The indicated input/output protocol is invalid. This error occurs when the IOPROTOCOL attribute is set to 1, but the front-end process does not support completion of outstanding READ requests as specified in the terminal control process's (TCP's) call to the CONTROL procedure.

Effect. The operation fails.

Recovery. Use the ALTER TERM command to specify a valid value for the IOPROTOCOL attribute. The INFO field of the STATUS TERM display shows the reply value received by the TCP from the PATHMON process.

3055

```
*3055* TERM term-name, INVALID I/O PROTOCOL VALUE FROM  
PATHMON (protocol-number)
```

Cause. The value specified for the IOPROTOCOL attribute, indicated by the protocol number, is invalid. The PATHMON process reports this error if the invalid value is greater than 15; otherwise, PATHCOM reports error 2011, ILLEGAL NUMBER.

Effect. The operation fails.

Recovery. Use the ALTER TERM command to specify a valid value for the IOPROTOCOL attribute. The INFO field of the STATUS TERM display shows the reply value received by the TCP from the PATHMON process.

3056

```
*3056* TERM term-name, SCREEN CACHE ERROR; TERMINAL SUSPENDED
```

Cause. A non-recoverable error occurred while a 6540 terminal was running in screen caching mode. Possible reasons for the error are memory parity errors and disk or diskette errors.

Effect. The terminal is suspended.

Recovery. Issue a RESUME TERM command to correct the situation; however, this procedure only succeeds if the 6540 device error is transient. If the error is not transient, run terminal diagnostics. If the RESUME TERM command does not correct the situation and terminal diagnostics do not reveal any problem, contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated

- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3057

```
*3057* TERM term-name, 6540 CACHE ERROR; REVERTING TO T16-6530 EMULATION
```

Cause. A 6540 terminal encountered an error while performing screen caching.

Effect. The caching function is disabled and the terminal reverts to 6530 emulation.

Recovery. Check the 6540 terminal for disk errors and other problems.

3058

```
*3058* TERM term-name, PROGRAM UNIT REQUIRES NEWER VERSION OF TCP
```

Cause. The SCREEN COBOL pseudocode for the program unit is not compatible with this version of the TCP.

Effect. The program terminates abnormally.

Recovery. Use the TCP that is at the same release level as the SCREEN COBOL compiler being used.

3059

```
*3059* TCP tcp-name, I/O ERROR ON DEVICEINFO (errnum)
```

Cause. A file-system error occurred on a call to the DEVICEINFO procedure.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3060

```
*3060* TERM term-name, DEVICE DOESN'T SUPPORT DOUBLEBYTE
CHARACTERS --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The terminal task that the TCP is attempting to link with has aborted for one of the following reasons:

- An attempt was made to run a SCREEN COBOL program unit containing double-byte characters on an IBM 3270 device or emulator that does not support Start Field Extended (SFE) orders.
- An attempt was made to run a SCREEN COBOL program unit containing double-byte characters on a 6530 device or emulator that does not support double-byte characters.
- An attempt was made to run a SCREEN COBOL program unit containing double-byte characters on an IBM 3270 device or emulator on an access method other than SNAX.

Effect. The program terminates abnormally.

Recovery. Abort the terminal from the PATHMON environment. Then either upgrade the terminal to support the application data or modify the requester-server application to conform to the device environment.

3061

```
*3061* TERM term-name, DBCS TRANSLATION SUPPORT NOT
INSTALLED --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. An attempt was made to run a SCREEN COBOL program unit containing double-byte characters on a system that does not have the correct set of procedures for the support of double-byte character sets.

Effect. The program terminates abnormally.

Recovery. Configure the Kanji translation library (T9115) using the SYSGEN phase of the Install program or the Distributed Systems Management/Software Configuration Manager (DSM/SCM) utility to accommodate Pathway/iTS terminals using double-byte character support.

3062

```
*3062* TERM term-name, INVALID KATAKANA OR DBCS DATA -
DESCRIPTOR (number) --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. Translation routines were unable to correctly translate data from internal format to the external format required by the device. The field in error is indicated by the DESCRIPTOR number.

Effect. The program terminates abnormally.

Recovery. Abort the terminal. Then either upgrade the terminal to support the application data or modify the requester-server application to conform to the device environment.

3063

```
*3063* TERM term-name, INVALID KATAKANA OR DBCS DATA -
INST ADDR inst-addr --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. Translation routines were unable to correctly translate data from internal format to the external format required by the device. The SCREEN COBOL operand that encountered the problem is indicated in the INST ADDR field.

Effect. The program terminates abnormally.

Recovery. Abort the terminal. Then either upgrade the terminal to support the application data or modify the requester-server application to conform to the device environment.

Debugging information: this error is the same as that reported by message 3062, except that, for internal reasons, the data descriptor is not available for reporting in the message. The offset can be mapped to the SCOBOLX verb that failed if the compiler ?MAP directive is used.

3064

```
*3064* TERM term-name, TRUNCATION OCCURRED DURING DISPLAY OF
DBCS DATA --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The TCP displayed a truncated data item. Truncation can occur with double-byte character data items that equal the maximum field length specified. A screen field containing double-byte characters can be truncated for one of the following reasons:

- Insertion of Shift In-Shift Out characters into a data stream being sent to an IBM device caused the number of displayable characters to exceed the field size.

- A double-byte character started on the last column of a Fujitsu 3270 device. As a result, a space was inserted into the data stream that caused the number of displayable characters to exceed the field size.
- A double-byte character started on the last column of a JET6530 or PCT emulator. As a result, a space was inserted into the data stream that caused the number of displayable characters to exceed the field size.

Effect. None.

Recovery. Truncation of double-byte data may be acceptable. If not, define larger data field lengths in the requester-server application, or define fields that do not wrap from one row to the next.

3065

```
*3065* TERM term-name, DEVICE DOES NOT SUPPORT KATAKANA -
DESCRIPTOR number --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The device that the terminal control process (TCP) attempted to link with does not support Katakana data.

Effect. The program terminates abnormally.

Recovery. Abort the terminal. Then either upgrade the terminal to support the application data or modify the requester-server application to conform to the device environment.

Debugging information: the descriptor number appears on the compiler MAP listing in the DESC column and indicates the ITEM NAME that caused the error.

3066

```
*3066* TERM term-name, DEVICE DOES NOT SUPPORT KATAKANA -
INST ADDR inst-addr --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The device that the terminal control process (TCP) attempted to link with does not support Katakana data.

Effect. The program terminates abnormally.

Recovery. Abort the terminal. Then either upgrade the terminal to support the application data or modify the requester-server application to conform to the device environment.

Debugging information: this error is the same as that reported by message 3065, except that, for internal reasons, the data descriptor is not available for reporting in the message. The offset can be mapped to the SCOBOLX verb that failed if the compiler ?MAP directive is used.

3067

```
*3067* TERM term-name, FIELD CONTAINS OTHER THAN DCBS -  
DESCRIPTOR number --- INST CODE: instruct AT OFFSET %offset  
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The TCP encountered a data field that did not contain only double-byte (Kanji) character data. The SCREEN COBOL programmer has attempted to display mixed or single-byte characters in a PIC N only screen field, or to display a mixed field on an IBM 3270 device that does not support mixed fields.

Effect. The program terminates abnormally.

Recovery. Abort the terminal. Then either upgrade the terminal to support the application data or modify the requester-server application to conform to the device environment.

Debugging information: the descriptor number appears on the compiler MAP listing in the DESC column and indicates the ITEM NAME that caused the error.

3068

```
*3068* FIELD CONTAINS OTHER THAN DBCS DATA -  
INST ADDR inst-addr --- INST CODE: instruct AT OFFSET %offset  
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The terminal control process (TCP) encountered a data field that does not contain *only* double-byte (Kanji) character data. The SCREEN COBOL programmer has attempted to display mixed or single-byte characters in a PIC N only screen field, or to display a mixed field on an IBM 3270 that does not support mixed fields. This error could also occur on 65xx terminals when the wrong TCP library is used.

Effect. The program terminates abnormally.

Recovery. Abort the terminal. Then, depending on the cause of the error, either upgrade the terminal to support the application data, modify the requester-server application to conform to the device environment, or use the correct TCP library.

Debugging information: this is the same error as message 3067, except that, for internal reasons, the data descriptor number is not available for reporting in the message.

3069

```
*3069* TERM term-name, UNILATERAL ABORT: BACKUP TASK STATE
NOT VALID --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT program-unit(version) IN TCLPROG tclprog
```

Cause. The terminal control process (TCP) encountered a backup open error for the identified TERM task. This message is generated when a TERM task fails and the backup TCP becomes the primary TCP before the TERM task is deleted.

Effect. The new primary terminal task is not in a valid executable state; the TERM task remains suspended.

Recovery. Abort and restart the TERM object.

3070

```
*3070* TERM term-name BACKUP TASK ERROR: UNABLE TO CHECKOPEN
TERMINAL --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. A TERM task has been suspended because the terminal control process (TCP) encountered a backup open error for the identified TERM task. This message is generated when the following sequence of events occurs:

1. The backup TCP marks the TERM task as suspended but does not delete it until the ABORT TERM command is executed.
2. The backup TCP sends a request to the primary TCP to suspend the corresponding primary TERM task.

Effect. The primary and backup terminal tasks terminate abnormally.

Recovery. Restart the TERM object either manually through PATHCOM or automatically by having the AUTORESTART attribute value greater than zero.

3071

```
*3071* TERM term-name REQUESTED DEVICE COLOR/HIGHLIGHT NOT
AVAILABLE --- INST CODE: instruct AT OFFSET %offset
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. The terminal does not have the attribute or color capability specified by the running SCREEN COBOL program unit.

Effect. The terminal terminates abnormally.

Recovery. Upgrade the terminal to support the application data or modify the requester-server application to conform to the device's limits.

3072

```
*3072* RUN-TIME DYNAMIC ATTRIBUTE SETTING INVALID ---  
INST CODE:  instruct AT OFFSET %offset  
IN PROGRAM UNIT unit(version) IN TCLPROG tclprog
```

Cause. An invalid attribute setting was detected, or the specified attribute setting would cause an out-of-bounds condition to occur during the run-time processing of the control structure.

Effect. The SCREEN COBOL program is suspended.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3073

```
*3073* INSUFFICIENT TERMPPOOL FOR REQUEST (bytes)
```

Cause. A SCREEN COBOL statement requires a larger terminal input buffer than is configured for the TERMPPOOL attribute. The buffer size requested is indicated in bytes.

Effect. The terminal task is suspended.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3074

```
*3074* ILLEGAL DELAY VALUE
```

Cause. An invalid DELAY variable timeout value was used.

Effect. The terminal task is suspended.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3100

```
*3100* TCP tcp-name, ERROR DURING SERVER OPEN (errnum)
```

Cause. A file-system error occurred while the specified terminal control process (TCP) was opening a server process.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3101

```
*3101* TCP tcp-name, TCLPROG DIRECTORY FILE OPEN ERROR  
(errnum)
```

Cause. A file-system error occurred while the terminal control process (TCP) was opening the TCLPROG directory file.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3102

```
*3102* TCP tcp-name, TCLPROG CODE FILE OPEN ERROR (errnum)
```

Cause. A file-system error occurred while the specified terminal control process (TCP) was opening the TCLPROG code file.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3103

```
*3103* TCP tcp-name, ILLEGAL TCLPROG DIRECTORY FILE
```

Cause. The format of the file specified as the TCLPROG directory file is invalid. This is an internal error.

Effect. The TCLPROG files cannot be used.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3104

```
*3104* TCP tcp-name, ILLEGAL TCLPROG CODE FILE
```

Cause. The format of the file specified as the TCLPROG code file is invalid. This is an internal error.

Effect. The TCLPROG files cannot be used.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3106

```
*3106* TCP tcp-name, PARAMETERS FOR TCP CONFIGURATION TOO  
LARGE
```

Cause. The terminal control process (TCP) configuration attributes are incorrect. This message is usually caused by incompatible values in the TCP configuration.

Effect. The TCP terminates.

Recovery. Use the INFO TCP command to check the values of the TERMBUF, TERMPPOOL, MAXTERMDATA, SERVERPOOL, MAXREPLY, MAXTERMS, MAXSERVERCLASSES, and MAXSERVERPROCESSES attributes.

Use the ALTER TCP command to correct the configuration.

3107

3107 TCP *tcp-name*, SWAP FILE CREATION ERROR (*errnum*)

Cause. A file-system error occurred while the specified terminal control process (TCP) was attempting to obtain swap space from the Kernel Managed Swap Facility (KMSF).

Effect. The swap space is not obtained from KMSF.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3108

3108 TERM *term-name*, SWAP FILE OPEN ERROR (*errnum*)

Cause. A file-system error occurred while the specified terminal control process (TCP) was opening a swap file.

Effect. The swap file is not opened.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3109

3109 TERM *term-name*, SWAP FILE I/O ERROR (*errnum*)

Cause. A file-system error occurred during an I/O operation to a swap file.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3110

3110 TERM *term-name*, NO ROOM FOR NEW SERVER CLASS IN TCP

Cause. The TCP attempted to communicate with more server classes than defined for the MAXSERVERCLASSES attribute.

Effect. The SEND operation to the server class fails.

Recovery. Redefine the value of the MAXSERVERCLASSES attribute or attempt the SEND operation again after a request in progress has completed.

3112

3112 TERM *term-name*, REPLY NUMBER NOT KNOWN TO PROGRAM

Cause. A server replied with a message containing a reply code not listed in the SEND statement of the program.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3113

3113 TERM *term-name*, TRANSACTION MESSAGE SIZE EXCEEDS LIMIT

Cause. The size of a message for a SEND to a server exceeds that allowed by the SERVERPOOL attribute in the terminal control process (TCP) configuration.

Effect. The operation fails.

Recovery. Redefine the value of the SERVERPOOL attribute, or isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3114

3114 TERM *term-name*, MAXIMUM REPLY SIZE EXCEEDS LIMIT

Cause. The size of either the largest SEND message or the longest possible reply exceeds the size defined for the MAXREPLY attribute.

Effect. The operation fails.

Recovery. Redefine the MAXREPLY attribute, or isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3115

3115 TERM *term-name*, TRANSACTION REPLY SIZE INVALID
(*bytes-length*)

Cause. The size of the reply received from the server did not correspond to the size of the selected YIELDS list given in the SEND statement of the program. *byte-length* is the length of the reply received in bytes.

If a reply exceeds the largest number of bytes allowed, the terminal control process (TCP) reports the byte count as the maximum reply length allowed plus 1.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3116

```
*3116* TCP tcp-name, ERROR DURING SERVER I/O (errnum)
```

Cause. A file-system error occurred during I/O to a server process.

Effect. The I/O operation fails. The terminal control process (TCP) unlinks from the server class.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3117

```
*3117* object-name SERVER CLASS UNDEFINED
```

Cause. The server class specified in a SEND statement is not defined for this PATHMON environment.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3118

```
*3118* TCP tcp-name, REQUEST INVALID FOR SERVER STATE
```

Cause. The state of the specified server does not allow the requested operation to occur. This is an internal error.

Effect. The operation fails.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version

- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3119

```
*3119* TCP tcp-name, NO SPACE FOR NEW SERVER PROCESS
```

Cause. The terminal control process (TCP) attempted to link with more server processes than are defined for the MAXSERVERPROCESSES attribute.

Effect. The operation fails.

Recovery. Redefine the MAXSERVERPROCESSES attribute and repeat the operation.

3121

```
*3121* TCP tcp-name, TCLPROG DIRECTORY FILE ERROR (errnum)
```

Cause. A file-system error occurred during an I/O operation to the TCLPROG directory file.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3122

```
*3122* TCP tcp-name, TCLPROG CODE FILE ERROR (errnum)
```

Cause. A file-system error occurred during an I/O operation to the TCLPROG code file.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3123

```
*3123* TERM term-name, NO SERVER PROCESS LINKED TO
```

Cause. The SCREEN COBOL run unit was waiting for a link to a server class when the server class became unavailable.

Effect. The SEND operation fails.

Recovery. Recovery depends on why the SEND failed, which is indicated in an earlier message. For additional information, see the description of the earlier message.

3124

```
*3124* TERM term-name, APPLICATION DEFINED ERROR--EXIT  
PROGRAM WITH ERROR (termination-status)
```

Cause. A SCREEN COBOL program executed an EXIT PROGRAM WITH ERROR verb with no higher-level CALL ON ERROR verb to handle the EXIT.

Effect. The program terminates. The value of the TERMINATION-STATUS special register is shown in the INFO field of the STATUS TERM display.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3125

```
*3125* TERM term-name, MULTIPLE UNSOLICITED MESSAGES REJECTED  
DUE TO TERM STOP/SUSPEND (unsol-msg-count)
```

Cause. The terminal control process (TCP) rejected unsolicited messages that were queued for a terminal when the terminal was stopped, suspended, or aborted, either programmatically or by an operator command.

Effect. The unsolicited messages are rejected.

Recovery. The INFO field of the STATUS TERM display specifies the number of unsolicited messages that were rejected. Restart the terminal and resend the unsolicited messages.

3140

```
*3140* TERM term-name, ERROR DURING I/O TO ROUTER (errnum)
```

Cause. A file-system error (*errnum*) occurred during an I/O operation to the router process performed by the terminal control process (TCP) on behalf of the specified TERM object.

Effect. The I/O operation fails and the TERM object is aborted.

Recovery. Make sure that the router process is running. Then stop and restart the TERM object.

3141

```
*3141* TERM term-name, SOCKET ERROR (tcPIP-errnum)
```

Cause. A TCP/IP file-system error (*tcPIP-errnum*) occurred during a socket function call performed by the terminal control process (TCP) on behalf of the specified TERM object.

Effect. The socket function call fails; however, the TERM object is not aborted or stopped. The TCP issues a new connection request to the router process.

Recovery. Informational message only; no corrective action is needed.

3142

```
*3142* TERM term-name, ROUTER CANNOT HANDLE MORE TCPS - LIMIT REACHED
```

Cause. A newly created terminal control process (TCP) sent its first “waiting for connection” request to the router process on behalf of the specified TERM object. However, the router could not handle the request because it was already servicing the maximum number of TCP processes (800).

Effect. All the TERM objects configured under the TCP that made the “waiting for connection” request are aborted.

Recovery. Change the FILE attribute of each of the suspended TERM objects to point to another router process. Then stop and restart the TERM objects.

3143

```
*3143* TERM term-name, ROUTER CANNOT HANDLE MORE TERMS - LIMIT REACHED
```

Cause. The sum of the number of TERM object requests for connection queued to the router process and the number of TERM objects currently connected to clients (that is, “in session”) has reached the maximum limit (32767).

Effect. The specified TERM object is aborted.

Recovery. Wait for a period of time (how long a time depends on the application) and then stop and restart the TERM object.

3144

3144 TERM *term-name*, MEMORY ALLOCATION ERROR IN ROUTER

Cause. A memory allocation error occurred within the router process while the router was maintaining connection request status information for the TERM objects.

Effect. The router process terminates abnormally. The TERM objects that were waiting for connections resend their requests to the router process.

Recovery. Restart the router process.

3145

3145 TERM *term-name*, ERROR WHILE CREATING SESSION

Cause. An error occurred when the terminal control process (TCP) attempted to create a session after establishing a socket connection with the client.

Effect. For the TERM object, the socket connection with the client is broken, the socket is closed, and a new “waiting for connection” request is made to the router process.

Recovery. Informational message only; no corrective action is needed.

3146

3146 TERM *term-name*, COMMUNICATION PROTOCOL IS NOT SUPPORTED

Cause. The communication protocol that the client used is not supported by Pathway/iTS. Pathway/iTS supports the HTTP and TCP/IP Raw Socket protocols.

Effect. The operation fails, the session is closed, and the TERM object is reinitialized for a new session.

Recovery. If the router process was not started with the *protocol* parameter set to HTTP or SOCKET, restart the router process with one of these protocol settings.

3147

3147 TERM *term-name*, SESSION IS NULL

Cause. The TCP/IP session associated with the socket connection was null when the terminal control process (TCP) either performed an I/O operation on the socket or attempted to delete it.

Effect. The operation (send to/receive from the client or session deletion) fails. The session is closed and the TERM object is reinitialized for a new session.

Recovery. Informational message only; no corrective action is needed.

3148

```
*3148* TERM term-name, ERROR DURING NOWAITED SOCKET SEND  
(tcPIP-errnum)
```

Cause. A TCP/IP file-system error (*tcPIP-errnum*) occurred during a nowait socket send operation to the client.

Effect. The send operation fails. The session is closed and the TERM object is reinitialized for a new session.

Recovery. Informational message only; no corrective action is needed.

3149

```
*3149* TERM term-name, ERROR IN XML GENERATION
```

Cause. The terminal control process (TCP) received an error in generating an Extended Markup Language (XML) document while sending a message to the browser-based client.

Effect. The send operation to the client fails. The session is closed and the TERM object is reinitialized for a new session.

Recovery. Informational message only; no corrective action is needed.

3150

```
*3150* TERM term-name, ERROR DURING NOWAITED SOCKET RECEIVE  
(tcPIP-errnum)
```

Cause. A TCP/IP file-system error (*tcPIP-errnum*) occurred during a nowait receive operation on a socket.

Effect. The receive operation fails. The session is closed and the TERM object is reinitialized for a new session.

Recovery. Informational message only; no corrective action is needed.

3161

```
*3161* TERM term-name, I/O ERROR (errnum)
```

Cause. A file-system error occurred during a SEND MESSAGE operation or during a REPLY TO UNSOLICITED MESSAGE operation.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3162

```
*3162* TERM term-name, RECEIVED MESSAGE SMALLER THAN EXPECTED  
(byte-length)
```

Cause. The message received from a RECEIVE UNSOLICITED MESSAGE operation, a TRANSFORM operation, or a SEND MESSAGE operation was smaller than expected. The length of the message received is indicated in bytes.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3163

```
*3163* TERM term-name, RECEIVED MESSAGE LARGER THAN EXPECTED  
(byte-length)
```

Cause. The message received from a RECEIVE UNSOLICITED MESSAGE operation or as a reply to a SEND MESSAGE operation was larger than expected. The length of the message received is indicated in bytes.

The number of bytes received by the terminal control process (TCP) during data overflow may be less than the number of bytes that the IDS or UMP process sends.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3164

```
*3164* TERM term-name, CODE OF RECEIVED MESSAGE UNDEFINED
```

Cause. The message received from a RECEIVE UNSOLICITED MESSAGE operation or as a reply to a SEND MESSAGE operation is undefined. The message code did not match any of the reply codes specified in the YIELDS or SELECT list.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3165

3165 TERM *term-name*, EDIT ERROR OCCURRED ON MESSAGE INPUT

Cause. An edit error occurred on input from an intelligent device.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1. If a USER CONVERSION clause is used in the program, check the user conversion routine.

3166

3166 TERM *term-name*, RECEIVED MESSAGE EXCEEDS MAXIMUM ALLOWABLE SIZE (*byte-length*)

Cause. The message received was larger than the maximum number of bytes allowed by the TCP configuration. The length of the received message is indicated in bytes.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3167

3167 TERM *term-name*, MESSAGE TO SEND EXCEEDS MAXIMUM ALLOWABLE SIZE (*byte-length*)

Cause. After conversion of message data from an unsupported format to a supported format, the message to be sent or received with a SEND MESSAGE statement was larger than the maximum number of bytes supported by the buffer. The length of the received message is indicated in bytes.

Effect. The operation fails.

Recovery. If the user conversion routine is causing the error, correct the routine. If the user conversion routine is not causing the problem, specify a larger reply area in the SEND MESSAGE statement in your program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3168

3168 TERM *term-name*, DEVICE SUBTYPE IS INVALID (*subtype*)

Cause. The device subtype specified for the TYPE attribute of an intelligent device is invalid.

Effect. The operation fails.

Recovery. Enter the correct subtype value (that is, 0, 1, or 2) for the intelligent device.

3169

3169 TERM *term-name*, ILLEGAL TIMEOUT VALUE (*timeout*)

Cause. The SCREEN COBOL program unit specified an invalid timeout value. The value was either less than 0 or could not be represented in an INT field.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3170

3170 TERM *term-name*, INVALID END OF MESSAGE CHARACTER
ENCOUNTERED

Cause. An invalid end-of-message character was encountered in an intelligent device message.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3171

3171 TERM *term-name*, FIELD LENGTH EXCEEDS MAXIMUM ALLOWABLE
LENGTH (*byte-length*)

Cause. The length of a field in an intelligent device message exceeded the maximum number of bytes allowed. *byte-length* represents the number of bytes in the field.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3172

```
*3172* TERM term-name, SEND MESSAGE LENGTH EXCEEDS MAXIMUM  
ALLOWED (byte-length)
```

Cause. After conversion of message data from an unsupported format to a supported format, the message length exceeded the maximum number of bytes allowed for all messages. The length of the message is indicated in bytes.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. If the user conversion routine is causing the error, correct the routine. If the user conversion routine is not causing the problem, specify a larger message area in the SEND MESSAGE statement in your program.

3173

```
*3173* TERM term-name, I/O ERROR ON CONTROL-26 OPERATION  
(errnum)
```

Cause. File-system error occurred during a CONTROL-26 operation to the terminal.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3174

```
*3174* TERM term-name, CONTROL-26 OPERATION DID NOT COMPLETE  
IN TIME
```

Cause. A timeout occurred during a CONTROL-26 operation to the terminal. The operation did not complete within 60 seconds.

Effect. The operation fails.

Recovery. If a front-end process is being used, verify that it handles CONTROL-26 operations properly. If not, a process may be hung, or there could be a system performance problem.

3175

3175 TERM *term-name*, EDIT ERROR OCCURRED ON MESSAGE OUTPUT

Cause. An edit error occurred on output to an intelligent device.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1. If a USER CONVERSION clause is used in the program, check the user conversion routine.

3176

3176 TERM *term-name*, ATTEMPT TO RECEIVE UNSOLICITED MESSAGE WITH ONE NOT YET REPLIED TO

Cause. A SCREEN COBOL program unit attempted to receive an unsolicited message before replying to a previously received unsolicited message.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1. On accepting an unsolicited message, a requester must always perform a REPLY TO UNSOLICITED MESSAGE statement before attempting to receive another unsolicited message.

3177

3177 TERM *term-name*, NO UNSOLICITED MESSAGE TO REPLY TO

Cause. A SCREEN COBOL program unit attempted to reply to an unsolicited message when there was no pending unsolicited message.

Effect. The reply attempt fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1. A requester must reply to an unsolicited message only after successfully receiving an unsolicited message.

3178

```
*3178* TERM term-name, ATTEMPT TO RECEIVE UNSOLICITED MESSAGE  
WHEN TERM MAXINPUTMSGs = 0
```

Cause. A SCREEN COBOL program unit attempted to receive an unsolicited message, but the terminal was not configured for unsolicited messages.

Effect. The operation fails.

Recovery. Configure the MAXINPUTMSGs attribute for the associated TERM or PROGRAM object to a value greater than zero.

3179

```
*3179* TERM term-name, DATA LEFT OVER ON SCATTER TO WORKING  
STORAGE
```

Cause. A SCREEN COBOL program unit attempted to execute the TRANSFORM verb, but there was data left unprocessed from the source data stream.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3180

```
*3180* TERM term-name, NOT ENOUGH DATA FOR SCATTER TO WORKING  
STORAGE
```

Cause. A SCREEN COBOL program unit tried to execute the TRANSFORM verb, but there was not enough data in the source data stream to satisfy the destination data items.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3181

```
*3181* TERM term-name, VARIABLE FIELD SIZE WOULD EXCEED  
DECLARED FIELD SIZE
```

Cause. A field being processed during delimited field processing is larger than the maximum size specified in its declaration.

Effect. The operation fails.

Recovery. This situation is disallowed only on output processing through the message section.

Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3182

<p>*3182* TERM <i>term-name</i>, DELIMITER IS NOT BYTE ALIGNED</p>
--

Cause. The delimiter's offset was not byte-aligned during delimited field processing. Therefore, the delimiter cannot be appended to the field in an outbound message or it cannot be found in the input data stream of an inbound message.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3183

<p>*3183* TERM <i>term-name</i>, DEPENDING VALUE IS OUT OF BOUNDS</p>

Cause. The value of the DEPENDING ON data item in the message section was larger than the limit specified in the OCCURS clause.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3184

<p>*3184* TERM <i>term-name</i>, CONFLICT OF DATA TYPES DURING 'PRESENT IF' DETERMINATION (<i>offset</i>)</p>

Cause. The control-field data type for PRESENT IF processing conflicts with the data supplied for its value. *offset* indicates the instruction address of the program unit.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3185

```
*3185* TERM term-name, FIELD OCCURRENCE EXCEEDS WORKING
STORAGE MAXIMUM OCCURRENCE (offset)
```

Cause. An error occurred during an attempt to map between working storage and the message section; the message section definition cannot support the structure in working storage. *offset* indicates the instruction address of the program unit.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3197

```
*3197* TCP tcp-name, TCP MEMORY DUMP NOT TAKEN (errnum)
```

Cause. The terminal control process (TCP) encountered the internal error *errnum* and its memory dump failed. *errnum* can have one of the following values:

- | | |
|-------------------|-------------------|
| 1 through 512 | File-system error |
| 1000 through 1512 | Fatal error |
| -1 through -9 | Special error |

Effect. The memory dump fails. If the error is a fatal error, the TCP terminates.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3200

3200 TCP *tcp-name*, INVALID FORMAT MESSAGE RECEIVED BY TCP

Cause. A terminal control process (TCP) or LINKMON process received an invalid request from the PATHMON process. This error usually occurs when incompatible versions of the PATHMON process and the TCP or the LINKMON process are running at the same time.

Effect. The request cannot be processed.

Recovery. Use versions of the PATHMON process and the TCP or LINKMON process that are compatible with each other. If the problem persists, contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3201

3201 TCP *tcp-name*, SYNCID VIOLATION IN MESSAGE RECEIVED BY TCP

Cause. A terminal control process (TCP) or LINKMON process received an invalid message sync ID. This is an internal error.

Effect. The request cannot be processed.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3202

3202 TCP *tcp-name*, TERMINAL IDENTIFIER NOT KNOWN TO TCP

Cause. The TCP encountered an unknown terminal identifier. This is an internal error.

Effect. The request cannot be processed.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3203

3203 TCP *tcp-name*, FUNCTION UNIMPLEMENTED

Cause. The TCP does not support the SCREEN COBOL function that was used.

Effect. The SCREEN COBOL program cannot be executed.

Recovery. Use a more recent version of the TCP.

3204

3204 TCP *tcp-name*, TCP STATE DOES NOT ALLOW OPERATION

Cause. The TCP is not in the appropriate state for the attempted operation.

This error is often issued after a STOP TCP command is attempted when the terminals are still running.

Effect. The operation fails.

Recovery. Repeat the operation when the TCP is in the appropriate state.

3205

3205 TERM *term-name*, TERMINAL STATE DOES NOT ALLOW OPERATION

Cause. The specified terminal is not in the appropriate state for the attempted operation.

Effect. The operation fails.

Recovery. Repeat the operation when the terminal is in the appropriate state.

3206

3206 TCP *tcp-name*, TCP CANNOT HANDLE MORE TERMINALS

Cause. The number of terminals required for the current operation exceeds that allowed by the MAXTERMS attribute in the terminal control process (TCP) configuration.

Effect. The operation fails.

Recovery. Increase the value of the MAXTERMS attribute and repeat the operation.

3207

3207 *object-name*, REQUEST PENDING

Cause. The request cannot be completed until the state of the specified object allows the request's completion. This message can be generated when a STOP SERVER command is issued before a FREEZE SERVER operation is completed. In such a case, the PATHMON process has not yet received messages from all its TCPs or LINKMON processes.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3208

3208 TCP *tcp-name*, NO ROOM FOR ANOTHER ALTERNATE TCLPROG

Cause. A total of 15 alternate TCLPROG file sets are already in use. The TCP can only support 16 distinct TCLPROG file sets, including its own and 15 other file sets specified in terminal configurations.

Effect. The operation fails.

Recovery. Use the SCREEN COBOL Utility Program (SCUP) to merge program units for all the requesters configured for the TCP into fewer than 16 pseudocode files; or, reassign requesters among TCPs so that the requesters assigned to a given TCP share fewer pseudocode files.

3209

```
*3209* TCP tcp-name, PROCESSOR DOES NOT HAVE PATHWAY  
MICROCODE
```

Cause. The microcode required to operate TS/MP or Pathway/iTS software is not installed on this system.

Effect. The TCP terminates.

Recovery. Install the appropriate microcode before starting the PATHMON environment.

3210

```
*3210* TCP tcp-name, BACKUP PROCESSOR DOWN
```

Cause. The backup CPU to the terminal control process (TCP) is down.

Effect. The TCP runs without a backup process.

Recovery. Reload the processor, or issue a CONTROL TCP, BACKUPCPU command to designate a new CPU for the TCP's backup process.

3211

```
*3211* TCP tcp-name, BACKUP PROCESS CREATION FAILURE :  
(process-creation-detail)
```

Cause. An operating system error occurred while attempting to create the TCP's backup process using the NEWPROCESS procedure. *process-creation-detail* provides more information about the failure.

Effect. The TCP executes without a backup.

Recovery. For additional information, see the description of the NEWPROCESS procedure in the *Guardian Procedure Calls Reference Manual*.

3212

```
*3212* TCP tcp-name, BACKUP FAILED
```

Cause. The backup process for the TCP has failed for an unknown reason.

Effect. The TCP recreates the backup process, if possible.

Recovery. If the problem persists, contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3213

```
*3213* TCP tcp-name, OPEN OF FILE TO BACKUP FAILED (errnum)
```

Cause. A file-system error occurred while the specified terminal control process (TCP) was trying to open a file to the backup process.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3214

```
*3214* TCP tcp-name, FILE-SYSTEM ERROR DURING CHECKPOINT (errnum)
```

Cause. A file-system error occurred during a CHECKPOINT operation.

Effect. The CHECKPOINT operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3215

```
*3215* TCP tcp-name, ERROR IN BACKUP: < nested message >
```

Cause. An error occurred in the backup process for the reason indicated in the nested message.

Effect. The backup process terminates.

Recovery. For additional information, see the description associated with the nested message.

3216

3216 TCP *tcp-name*, TCP INTERNAL ERROR (%*p-register*)

Cause. An internal TCP failure occurred. %*p-register* is the value of the P-register at the time of the failure, in octal.

Effect. The TCP terminates. If the DUMP attribute is on, the TCP takes a memory dump before terminating.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3217

3217 TCP *tcp-name*, TCP TRAP (%*p-register*)

Cause. A software or hardware failure resulted in a terminal control process (TCP) trap. This is an internal error. The value of the P register at the time of the failure is indicated.

Effect. The TCP terminates.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version

- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3218

```
*3218* TCP tcp-name, TAKEOVER BY BACKUP
```

Cause. The backup process took over processing from the primary process.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3219

```
*3219* TCP tcp-name, TCP MEMORY DUMP TAKEN - file-name
```

Cause. A terminal control process (TCP) memory dump in the specified file has completed.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3220

```
*3220* TCP tcp-name, INSPECT NOT ENABLED FOR TCP
```

Cause. The TERM is configured with the INSPECT attribute set to ON, but the TCP is configured with the INSPECT attribute set to OFF.

Effect. The HP *Inspect* symbolic debugger operation fails.

Recovery. Reconfigure the TCP with the INSPECT attribute set to ON.

3221

```
*3221* TCP tcp-name, INSPECT TERMINAL TABLE FULL
```

Cause. More than eight terminals per TCP attempted to use the Inspect process.

Effect. The Inspect operation fails.

Recovery. Reduce the number of terminals under the control of a TCP that are attempting to use the Inspect process.

3222

```
*3222* TCP tcp-name, INSPECT BREAKPOINT TABLE FULL
```

Cause. An attempt was made to set more than 20 program breakpoints for each TCP.

Effect. The Inspect operation fails.

Recovery. Set fewer than 20 program breakpoints for each TCP.

3223

```
*3223* TCP tcp-name, REQUEST NOT ALLOWED WHILE AT BREAKPOINT
```

Cause. An Inspect command was issued to a TERM already at a program breakpoint.

Effect. The Inspect operation fails.

Recovery. Resubmit the Inspect command when the TERM is not at a program breakpoint.

3224

```
*3224* I/O ERROR WITH IMON OR INSPECT (errnum)
```

Cause. The file-system error *errnum* occurred when the TCP was unable to complete a request to IMON or to start an Inspect process.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3225

```
*3225* TCP tcp-name, TASK ALREADY USING ANOTHER INSPECT  
TERMINAL
```

Cause. The file name specified in an Inspect command names a terminal different from the terminal currently used for debugging the SCREEN COBOL program.

Effect. The operation fails.

Recovery. Reissue the Inspect command using a valid file name.

3226

3226 TCP *tcp-name*, REQUESTED FUNCTION NOT SUPPORTED IN THIS RELEASE

Cause. The PATHMON process requested a function that is not supported in this version of the terminal control process (TCP) or LINKMON process.

Effect. The operation fails.

Recovery. Use a newer version of the TCP or LINKMON process.

3227

3227 TERM *term-name*, SEND: EXTERNAL PATHMON NAME INVALID

Cause. The PATHMON process name given in a SCREEN COBOL SEND statement is not in the correct form.

Effect. The operation fails.

Recovery. Check that the first character is a dollar sign (\$). Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3228

3228 TERM *term-name*, SEND: EXTERNAL SYSTEM NAME INVALID

Cause. The system name given in a SCREEN COBOL SEND statement is not in the correct form.

Effect. The operation fails.

Recovery. Check that the first character is a backslash (\). Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3229

3229 TERM *term-name*, SEND: EXTERNAL SYSTEM NAME NOT DEFINED

Cause. The SCREEN COBOL SEND statement does not contain a system name or the system name is not known to the network.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3230

```
*3230* TERM term-name, SEND: NO ROOM FOR NEW EXTERNAL PATHMON  
IN TCP
```

Cause. A request required the terminal control process (TCP) to contact an external PATHMON process, but the number of external PATHMON processes had already reached the value specified for the MAXPATHWAYS attribute in the TCP configuration.

Effect. The operation fails.

Recovery. Try the SEND operation again or increase the value defined for the MAXPATHWAYS attribute.

3231

```
*3231* TERM term-name, SEND: ERROR DURING I/O TO EXTERNAL  
PATHMON (errnum)
```

Cause. A file-system error occurred during an OPEN or WRITEREAD operation to an external PATHMON process.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3232

```
*3232* TCP tcp-name, MAXTERMDATA IN TCP CONFIGURATION TOO  
SMALL
```

Cause. The terminal control process (TCP) received a request that required more data space than what is defined for the MAXTERMDATA attribute.

Effect. The operation fails.

Recovery. Increase the value of the MAXTERMDATA attribute or reduce the size of the request.

3233

3233 TCP *tcp-name*, SERVER PROCESS UNKNOWN

Cause. The terminal control process (TCP) or LINKMON process received a message from the PATHMON process containing a reference to a server process that is unknown to the TCP or LINKMON process. This is an internal error.

Effect. The TCP or LINKMON process cannot process the request.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered
- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3235

3235 TCP *tcp-name*, CODE AREA TOO SMALL - TCP THRASHING

Cause. The terminal control process (TCP) is replacing excessive amounts of code because it cannot find space for new programs.

Effect. The TCP functions inefficiently.

Recovery. Increase the value of the CODEAREALEN attribute in the TCP configuration.

3237

3237 TCP *tcp-name*, TCP EXPERIENCING PERSISTENT FILE I/O ERROR

Cause. The terminal control process (TCP) encountered persistent input/output (I/O) errors while attempting to access the TCLPROG files. The errors persisted for 100 to 500 retries, with a 1-second delay for each retry.

Effect. The TCP terminates.

Recovery. Check the log file for the name of the file that is causing the problem; then correct the problem.

3238

```
*3238* TERM term-name, PARAMETER REFERENCE EXCEEDS FORMAL  
PARAMETER SIZE
```

Cause. A linkage section in a called program is larger than the maximum size of the corresponding parameter in the calling program.

Effect. The call fails.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. For instructions on how to locate the problem within the program, see [General Information](#) on page 13-1.

3239

```
*3239* TCP tcp-name, GUARDIAN-LIB INCOMPATIBLE WITH TCP
```

Cause. The object file of the terminal control process (TCP) program is incompatible with its user-library object file.

Effect. The TCP terminates.

Recovery. Use compatible versions of the PATHTCP2 program object file and the user-library file.

3240

```
*3240* TCP tcp-name, VALUE FOR MAXINPUTMSGs TOO LARGE
```

Cause. The value specified for the MAXINPUTMSGs attribute for a TCP object exceeded 2045. This is a TCP internal error, because PATHCOM and the PATHMON process do not accept MAXINPUTMSGs values greater than 2045.

Effect. The TCP terminates.

Recovery. Contact your service provider and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the message(s) generated
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

- Supporting documentation such as Event Management Service (EMS) logs, trace files, and dump files, if applicable

If your local operating procedures require contacting the GCSC, supply your system number and the numbers and versions of all related products as well.

3241

```
*3241* TCP tcp-name, UNSOLICITED MESSAGE REJECTED BY TCP
(errnum)
```

or

```
*3241* TERM term-name, UNSOLICITED MESSAGE REJECTED BY TCP
(errnum)
```

Cause. The terminal control process (TCP) rejected an unsolicited message for the reason indicated by the error number. (This error number is also shown in the INFO field of the STATUS TERM display.) This message can be generated on behalf of either an individual requester or the entire TCP.

The meaning of the error number is as follows:

Error Number	Description
0	No error
1	A file-system error occurred.
2	The terminal is suspended.
3	The terminal queue is full. (The TERM MAXINPUTMSGSGS configuration value has been exceeded.)
4	The TCP queue area is full.
5	Queuing is not enabled. (The TCP MAXINPUTMSGSGS configuration value is set to 0.)
6	The length field in the message exceeds the configured maximum for the TCP.
7	Unrecognizable request code (refers to MSG-ID and MSG-VERSION fields).
8	The terminal was not found.
9	The MAXINPUTMSGSGS value has been exceeded.
10	A RECEIVE UNSOLICITED MESSAGE error occurred. INFO1 provides the reason, which is the value obtained from the TERMINATION-STATUS register.
11	The number of bytes in the reply message sent by the SCREEN COBOL requester exceeded the number of bytes supported by the sender of the unsolicited message. This is a warning and should occur only during the application-debugging phase.

Effect. The TCP rejects the unsolicited message and replies to the sender without delivering it to the target requester.

Recovery. Take recovery action based on the error number. For information about unsolicited-message errors, see the *HP NonStop Pathway/iTS TCP and Terminal Programming Guide*.

3242

```
*3242* TCP tcp-name, MULTIPLE UNSOLICITED MESSAGES REJECTED
```

Cause. This message is returned only if unsolicited messages are arriving and being rejected with sufficient frequency that the terminal control process (TCP) cannot log individual error messages for each rejected unsolicited message.

Effect. The TCP rejected one or more unsolicited messages and replied to their sender without delivering them to their target requester.

Recovery. The processes that sent the rejected messages should perform recovery action on receiving this message.

3243

```
*3243* TCP tcp-name, PATHTCP2 CANNOT EXECUTE ON THIS RELEASE  
OF THE OPERATING SYSTEM
```

Cause. The TCP cannot execute on the currently running version of the operating system.

Effect. The TCP stops.

Recovery. Use compatible versions of the operating system and the PATHTCP2 process.

3244

```
*3244* TCP tcp-name, BACKUP STARTED
```

Cause. A backup TCP process has been started and is ready to take over for the primary TCP process, if necessary.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

3245

```
*3245* SEND: Pathway/TS functionality not licensed
```

Cause. The external Pathmon process rejects the OPEN message due to licensing problems.

Effect. The operation fails.

Recovery. A licensed copy of Pathway/iTS should be installed.

3246

3246 SERVER CLASS IS EITHER IN FREEZE PENDING OR IN FROZEN STATE

Note. This error does not appear in the RETCODE token or in an error list; it appears only in response to the STATUS TCP or STATUS TERM command.

Cause. The server class to which a send or a dialog was targeted, is either in freeze-pending or in frozen state.

Effect. The operation fails. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in the SCREEN COBOL program. If ON ERROR clause is not mentioned, then TCP suspends the terminal after logging the error against the term; however, the term can be resumed to restart processing at the last verb.

Recovery. Isolate the server class within the SCREEN COBOL program and make sure the server class is in STARTED mode before applying SEND/DIALOG to it. For instructions on how to isolate the server class within the program, see [Additional Error Information](#) on page 13-1

3247

3247 A NON-RETRYABLE LINKMON CONNECT ERROR OCCURED (errnum)

Note. This error does not appear in the RETCODE token or in an error list; it appears only in response to the STATUS TCP or STATUS TERM command.

Cause. A non-retryable LINKMON connect error occurred on both primary and backup CPU of the TCP. The associated file error gives the file-system error occurred with the LINKMON process.

Effect. The operation fails. TCP, after reporting the error, aborts the terminal irrespective of ON ERROR clause mentioned or not.

Recovery. Although other file-system errors are also possible, most typical file-system errors are error 14 (FENOSUCHDEV) and 201 (FEPATHDOWN). File-system error 14 indicates that the LINKMON process is not up on the CPU. File-system error 201 indicates that the LINKMON process had abended due to some reason. Correct the LINKMON problem and then restart the term.

3248

3248 A RETRYABLE LINKMON CONNECT ERROR OCCURRED (errnum)

Note. This error does not appear in the RETCODE token or in an error list; it appears only in response to the STATUS TCP or STATUS TERM command.

Cause. A retryable LINKMON connect error occurred on primary CPU of the TCP. The associated file error gives the file-system error occurred with the LINKMON process. Typical file-system errors are 100 (FENOTRDY), 28(FETOOMANY) and 30 (FENOLCB). Here are the possible causes for this error.

- The LINKMON process on the CPU is just started and is not yet initialized (Error #100).
- More than 255 sends are being attempted by the same TCP process (Error #28).
- More than 4K sends are being attempted on the same CPU (Error #30).

Effect. The operation fails. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in the SCREEN COBOL program. If ON ERROR clause is not mentioned, then TCP suspends the terminal after logging the error against the term; however, the term can be resumed to restart processing at the last verb.

Recovery. If the associated file-system error is 100 then it means that the LINKMON process is just started on the CPU and is not yet ready to accept the Pathsend requests. The term should retry the operation after a time delay. If associated file-system error is 30, then it means that the outstanding incoming requests to the LINKMON process have been reached. This is also a transient error and the term should retry the operation.

3249

3249 TCP FAILED WHEN A SEND/DIALOG TO SERVER WAS
OUTSTANDING

Note. This error does not appear in the RETCODE token or in an error list; it appears only in response to the STATUS TCP or STATUS TERM command.

Cause. TCP abended while a SEND or a DIALOG I/O to server process was outstanding and the term was not under transaction.

Effect. The operation fails. The server might or might not have received the message. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in the SCREEN COBOL

program. If ON ERROR clause is not mentioned, then TCP aborts the terminal after logging the error against the term. If the term is associated with any dialog, the dialog is also aborted.

Recovery. Recovery is application specific. Since TCP does not know the status of the last I/O hence the SEND/DIALOG operation can be retried if it is OK to send duplicate message to the server class. Pathway/iTS 1.1 strongly recommends that the send and dialog operations to server classes that don't allow duplicate messages must be wrapped under TMF transactions.

3250

3250 DIALOG MODE VIOLATION

Note. This error does not appear in the RETCODE token or in an error list; it appears only in response to the STATUS TCP or STATUS TERM command.

Cause. The context sensitive programming protocol has been violated by the SCREEN COBOL program. The possible causes of this error are:

- A second DIALOG-BEGIN was attempted when the term is already under active dialog.
- A DIALOG-SEND was attempted without the term being under active dialog.
- A DIALOG-END was attempted without the term being under active dialog.
- A DIALOG-ABORT was attempted without the term being under active dialog.

Effect. The operation fails. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in the SCREEN COBOL program. If ON ERROR clause is not mentioned, then TCP aborts the terminal after logging the error against the term. If the term is associated with any dialog the dialog is also aborted.

Recovery. There is no recovery action on this. Correct the SCREEN COBOL program and restart the term. For instructions on how to isolate the problem within the program, see Additional Information for Terminal Errors at the beginning of this section.

3251

3251 DIALOG HAS NOT BEEN ENDED BY THE SERVER

Note. This error does not appear in the RETCODE token or in an error list; it appears only in response to the STATUS TCP or STATUS TERM command.

Cause. A DIALOG-END statement is called when the server has not yet ended the dialog.

Effect. The operation fails. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in the SCREEN COBOL program. If ON ERROR clause is not mentioned, then TCP aborts the terminal after logging the error against the term. If the term is associated with any dialog the dialog is also aborted.

Recovery. There is no recovery action on this. Correct the SCREEN COBOL program to make sure that the DIALOG-END is called only after the server has ended the dialog. For instructions on how to isolate the problem within the program, see Additional Information for Terminal Errors at the beginning of this section.

3252

3252 DIALOG HAS BECOME INVALID DUE TO A PRIMARY TCP FAILURE

Cause. A DIALOG-SEND, DIALOG-END or DIALOG-ABORT statement is called and it was found that the dialog has been aborted by the LINKMON process due to the primary TCP failure.

Effect. The operation fails. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in the SCREEN COBOL program. If ON ERROR clause is not mentioned, then TCP aborts the terminal after logging the error against the term.

Recovery. Clean up the requester portion of dialog by calling DIALOG-END and initiate a new dialog if required.

3253

3253 ERROR OCCURRED DURING DIALOG VERB EXECUTION (errnum)

Cause. A dialog statement has hit a dialog related error. Possible causes of this error are:

- A DIALOG-BEGIN or DIALOG-SEND statement failed because either the server requested a dialog abort or the server abended before replying to the transaction request.
- A DIALOG-SEND statement failed because the dialog has already been ended or aborted.
- A DIALOG-BEGIN or DIALOG-SEND statement failed because the associated transaction has been aborted which caused the dialog to be aborted as well.

Effect. The operation fails. If ON ERROR clause is mentioned, then error is reported against the term but the term continues to run depending on the error-handling logic in

the SCREEN COBOL program. If ON ERROR clause is not mentioned, then TCP aborts the terminal after logging the error against the term.

Recovery. Clean up the requester portion of dialog by calling DIALOG-END and initiate a new dialog if required.

A Syntax Summary

This appendix contains summaries of the PATHCOM commands described in this manual. All commands are listed in alphabetical order. Note that PATHCOM commands that create or manage TS/MP objects, such as SERVER and LINKMON objects, are described in the *TS/MP System Management Manual*.

```
ABORT { [ TERM ] term-name
        { [ TERM ] ( term-name [ , term-name ]... )
        { TERM * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
TCP tcp-name
```

```
ADD PROGRAM program-name [ , program-attribute ]...
```

```
ADD TCP tcp-name [ , tcp-attribute ]...
```

```
ADD TERM term-name [ , term-attribute ]...
```

```
ALTER [ PROGRAM ] program-name
```

```
{ , program-attribute
{ , RESET program-keyword
{ , RESET (program-keyword [ , program-keyword ]... ) }
```

program-keyword is:

```
ERROR-ABORT
OWNER
PRINTER
SECURITY
TCP
TMF
TYPE program-type
```

```
ALTER [ TCP ] tcp-name
```

```
{ , tcp-attribute }
{ , RESET tcp-keyword }
{ , RESET ( tcp-keyword [, tcp-keyword ]... ) }
```

tcp-keyword is:

AUTORESTART	INSPECT	POWERONRECOVERY
CHECK-DIRECTORY	MAXINPUTMSGLEN	PRI
CODEAREALEN	MAXINPUTMSG	PROCESS
CPU	MAXPATHWAYS	PROGRAM
DEBUG	MAXREPLY	SERVERPOOL
DUMP	MAXSERVERCLASSES	SENDMSGTIMEOUT
GUARDIAN-LIB	MAXSERVERPROCESSES	STATS
GUARDIAN-SWAP	MAXTERMDATA	SWAP
HIGHPIN	MAXTERMS	TCLPROG
HOMETERM	NONSTOP	TERMBUF
		TERMPPOOL

```
ALTER [ TERM ] term-name
```

```
{ , term-attribute [, term-attribute ]... }
{ , RESET term-keyword }
{ , RESET ( term-keyword [, term-keyword ]... ) }
```

term-keyword is:

AUTORESTART	FILE	TCLPROG
BREAK	INITIAL	TCP
DIAGNOSTIC	INSPECT	TMF
DISPLAY-PAGES	IOPROTOCOL	TRAILINGBLANKS
ECHO	MAXINPUTMSG	TYPE
EXCLUSIVE	PRINTER	

```
CONTROL [ TCP ] { tcp-name }
                  { ( tcp-name [, tcp-name ]... ) }
                  { tcp-attribute [, tcp-attribute ]... }
```

tcp-attribute is:

```
BACKUPCPU number
DUMP { ON [ ( FILE file-name ) ] | OFF }
DUMPMEMORY { PRIMARY | BACKUP | BOTH } [ ( FILE file-name ) ]
STATS { ON | OFF }
```

```
DELETE [ PROGRAM ] { program-name
                     { ( program-name [ , program-name ]... ) } }
```

```
DELETE [ TCP ] { tcp-name
                 { ( tcp-name [ , tcp-name ]... ) } }
```

```
DELETE TELL number
```

```
DELETE [ TERM ] { term-name
                  { ( term-name [ , term-name ]... ) } }
```

```
INFO [ / OUT list-file / ]
    { [ PROGRAM ] program-name
      { [ PROGRAM ] ( program-name [ , program-name ]... ) }
      PROGRAM *
    }
    [ , OBEYFORM ]
```

```
INFO [ / OUT list-file / ]
    { [ TCP ] tcp-name
      { [ TCP ] ( tcp-name [ , tcp-name ]... ) }
      TCP * [ , option [ , option ] ]
    }
    [ , OBEYFORM ]
```

option is:

```
STATE state-type
OBEYFORM
```

```
INFO [ / OUT list-file / ] TELL { number
                                   { *
                                   }
```

```
INFO [ / OUT list-file / ]
```

```

{ [ TERM ] term-name [ , OBEYFORM ] }
{ [ TERM ] (term-name [ , term-name ]...) [ , OBEYFORM ] }
{ TERM * [ , option [ , option ]... ] }
```

option is:

```

STATE state-type
TCP tcp-name
OBEYFORM
```

```
INSPECT [ TERM ] term-name [ , FILE file-name ]
```

```

PRIMARY { [ TCP ] { tcp-name
                  { (tcp-name [ , tcp-name ] ... ) } }
          { TCP *
            [ , IFPRICPU number ] }
```

```

REFRESH-CODE { [ TCP ] tcp-name
               { [ TCP ] ( tcp-name [ , tcp-name ]... ) }
               { TCP * [ , STATE state-type ] }
```

```
RESET PROGRAM [ program-keyword [ , program-keyword ]... ]
```

program-keyword is:

```

ERROR-ABORT
OWNER
PRINTER
SECURITY
TCP
TMF
TYPE program-type
```

```
RESET TCP [ tcp-keyword [ , tcp-keyword ]... ]
```

tcp-keyword is:

AUTORESTART	INSPECT	POWERONRECOVERY
CHECK-DIRECTORY	MAXINPUTMSGLEN	PRI
CODEAREALEN	MAXINPUTMSG	PROCESS
CPUS	MAXPATHWAYS	PROGRAM
DEBUG	MAXREPLY	SENDMSGTIMEOUT
DUMP	MAXSERVERCLASSES	SERVERPOOL
GUARDIAN-LIB	MAXSERVERPROCESSES	STATS
GUARDIAN-SWAP	MAXTERMDATA	SWAP
HIGHPIN	MAXTERMS	TCLPROG
HOMETERM	NONSTOP	TERMBUF
		TERMPPOOL

```
RESET TERM [ term-keyword [ , term-keyword ]... ]
```

term-keyword is:

AUTORESTART	FILE	TCLPROG
BREAK	INITIAL	TCP
DIAGNOSTIC	INSPECT	TMF
DISPLAY-PAGES	IOPROTOCOL	TRAILINGBLANKS
ECHO	MAXINPUTMSG	TYPE
EXCLUSIVE	PRINTER	

```
RESUME { [ TERM ] term-name
        { [ TERM ] ( term-name [ , term-name ]... ) }
        { TERM * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
TCP tcp-name
```

```
RUN [ PROGRAM ] program-name
    [ , program-parameter [ , program-parameter ]... ]
```

program-parameter is:

```
    PRINTER { file-name [ IS-ATTACHED ] }
             { IS-ATTACHED }
    FILE file-name
    TYPE program-type
    NOWAIT
```

```
SET PATHWAY pw-attribute [ , pw-attribute ]...
```

pw-attribute is:

```
    MAXTCPS number
    MAXTERMS number
    MAXEXTERNALTCPS number
    MAXPROGRAMS number
    MAXTELLQUEUE number
    MAXTELLS number
    MAXTMFRESTARTS number
```



```
SET PROGRAM program-attribute [ , program-attribute ]...
```

program-attribute is:

```
TCP tcp-name
TYPE program-type (      INITIAL program-unit-name
                        [ , BREAK { ON | OFF }          ]
                        [ , DISPLAY-PAGES number        ]
                        [ , ECHO { ON | OFF | CURRENT }  ]
                        [ , EXCLUSIVE { ON | OFF }       ]
                        [ , IOPROTOCOL { 0 | 1 }         ]
                        [ , MAXINPUTMSGs number         ]
                        [ , TCLPROG file-name           ]
                        [ , TRAILINGBLANKS { ON | OFF } ] ) ...

]

ERROR-ABORT { ON | OFF }
LIKE program-name
OWNER owner-id
PRINTER { file-name [ IS-ATTACHED ] }
        { IS-ATTACHED }
SECURITY security-attribute
TMF { ON | OFF }
```

```
SET TCP tcp-attribute [ , tcp-attribute ]...
```

tcp-attribute is:

```

MAXTERMS number
TCLPROG file-name
AUTORESTART number
CHECK-DIRECTORY { ON | OFF }
CODEAREALEN double-number
CPUS primary : backup
DEBUG { ON | OFF }
DUMP { ON [ ( FILE file-name ) ] | OFF }
GUARDIAN-LIB file-name
GUARDIAN-SWAP $volume
HIGHPIN { ON | OFF }
HOMETERM file-name
INSPECT { ON [ ( FILE file-name ) ] | OFF }
LIKE tcp-name
MAXINPUTMSGLEN number
MAXINPUTMSGGS number
MAXPATHWAYS number
MAXREPLY bytes
MAXSERVERCLASSES number
MAXSERVERPROCESSES number
MAXTERMDATA bytes
NONSTOP { 0 | 1 }
POWERONRECOVERY { ON | OFF }
PRI priority
PROCESS $process-name
PROGRAM file-name
SENDMSGTIMEOUT { ON | OFF }
SERVERPOOL bytes
STATS { ON | OFF }
SWAP $volume
TERMBUF bytes
TERMPPOOL bytes

```

```
SET TERM term-attribute [ , term-attribute ]...
```

term-attribute is:

```
FILE file-name
INITIAL program-unit-name
TCP tcp-name
AUTORESTART number
BREAK { ON | OFF }
DIAGNOSTIC { ON | OFF }
DISPLAY-PAGES number
ECHO { ON | OFF | CURRENT }
EXCLUSIVE { ON | OFF }
INSPECT { ON [ ( FILE file-name ) ] | OFF }
IOPROTOCOL { 0 | 1 }
LIKE term-name
MAXINPUTMSGS number
PRINTER { file-name [ IS-ATTACHED ] }
        { IS-ATTACHED }
TCLPROG file-name
TMF { ON | OFF }
TRAILINGBLANKS { ON | OFF }
TYPE term-class
```

```
SHOW [ / OUT list-file / ] { PROGRAM
                             TCP
                             TERM }
```

```
START { [ TCP ] tcp-name
        { [ TCP ] ( tcp-name [ , tcp-name ]... ) }
        { TCP * [ , STATE state-type ] }
```

```
START { [ TERM ] term-name
        [ , INITIAL program-unit-name ] }
        { [ TERM ] ( term-name [ , term-name ]... )
        [ , INITIAL program-unit-name ] }
        { TERM * [ , option [ , option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
INITIAL program-unit-name
```

```

STATS [ / OUT list-file / ]

{ [ TCP ] tcp-name }
{ [ TCP ] ( tcp-name [ , tcp-name ]... ) }
{ TCP * [ , option [ , option ]... ] }

[ , tcp-attribute [ , tcp-attribute ]... ]

```

option is:

```

STATE state-type
COUNT number
DETAIL
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET

```

tcp-attribute is:

```

COUNT number
DETAIL
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET

```

```

STATS [ / OUT list-file / ]

    { [ TERM ] term-name
      [ , term-attribute [ , term-attribute ]... ]
    }
    { [ TERM ] ( term-name [ , term-name ]... )
      [ , term-attribute [ , term-attribute ]... ]
    }
    { TERM * [ , option [ , option ] ... ]
    }

```

term-attribute is:

```

COUNT number
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET

```

option is:

```

STATE state-type
TCP tcp-name
COUNT number
FREQTABLE
INTERVAL number { HRS | MINS | SECS }
RESET

```

```

STATUS [ / OUT list-file / ]

    { [ TCP ] tcp-name [ , DETAIL ] }
    { [ TCP ] ( tcp-name [ , tcp-name ]... ) [ , DETAIL ] }
    { [ TCP ] E\node.$process-name [ , DETAIL ] }
    { TCP * [ , option [ , option ] ] }

```

option is:

```

STATE state-type
DETAIL

```

```
STATUS [ / OUT list-file / ]
```

```
{ [ TERM ] term-name [ , DETAIL ] }
{ [ TERM ] ( term-name [ , term-name ]... ) [ , DETAIL ] }
{ TERM * [ , option [ , option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
DETAIL
```

```
STOP { [ TCP ] tcp-name [ , WAIT ] }
      { [ TCP ] ( tcp-name [ , tcp-name ]... ) [ , WAIT ] }
      { TCP * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
WAIT
```

```
STOP { [ TERM ] term-name }
      { [ TERM ] ( term-name [ , term-name ]... ) }
      { TERM * [ , option [ , option ] ] }
```

option is:

```
STATE state-type
TCP tcp-name
```

```
SUSPEND { [ TERM ] term-name [!] }
          { [ TERM ] ( term-name [ , term-name ]... ) }
          { TERM * [ option [ option ]... ] }
```

option is:

```
STATE state-type
TCP tcp-name
!
```

```
SWITCH { [ TCP ] tcp-name
        { [ TCP ] tcp-name [ , tcp-name ]... }
        { TCP * [ , STATE state-type ] }
```

```
TELL { [ TERM ] termname-list
      { TERM * [ , option [ , option ] ] } , "string"
```

option is:

```
STATE state-type
TCP tcp-name
```


B PATHCOM Reserved Words

This appendix contains a list of words that are reserved and should not be used for variable names in PATHCOM commands. Although it might be possible in some contexts to use words on this list, HP strongly recommends that these words not be used in any user-assigned names.

TS/MP Objects

The reserved words relating to TS/MP (server-oriented) objects, and the PATHMON and PATHWAY objects, appear in the following table:

ADD	DEBUG	HELP
ALTER	DEFINE	HIGHPIN
ALL	DELETE	HOMETERM
ARGLIST	DELETE-DELAY	HRS
ASSIGN	DETAIL	
ASSOCIATIVE	DUMP	I-O
AUTORESTART	DUMPMEMORY	IBM-3270
		IFPRICPU
BACKUP	ENV	IN
BACKUPCPU	ERRORS	INFO
BLOCK	EVENTFORMAT	INPUT
BOTH	EXCLUSIVE	INTELLIGENT
	EXIT	INTERVAL
CMDCWD	EXT	
CMDVOL		LIKE
CODE	FC	LINKDEPTH
COLD	FILE	LINKMON
COMMANDS	FREEZE	LOG1
CONTROL	FREQTABLE	LOG2
CONVERSATIONAL		
COOL	GUARDIAN	MAXASSIGNS
COUNT	GUARDIAN-LIB	MAXDEFINES
CPUS	GUARDIAN-SWAP	MAXEXTERNALTCPS
CREATEDELAY		MAXLINKS
CWD		MAXLINKMONS
MAXPARAMS	PROCESS	TMF
MAXPATHCOMS	PROCESSES	
MAXPROGRAMS	PROCESSTYPE	UNTIL

MAXSERVERCLASSES	PROGRAM	
MAXSERVERS	PROTECTED	VOLUME
MAXSPI		
MAXSTARTUPS	REC	WAIT
MAXTCPS	RESET	WARM
MAXTELLQUEUE		
MAXTELLS	SECS	
MAXTERMS	SECURITY	
MAXTMFRESTARTS	SEL	
MINS	SERVER	
MODE	SET	
	SHARED	
NODEINDEPENDENT	SHOW	
NUMSTATIC	SHUTDOWN	
	SHUTDOWN2	
OBEY	START	
OBEYFORM	STARTUP	
OBEYVOL	STATS	
OFF	STATUS	
ON	STDERR	
OPEN	STDIN	
OSS	STDOUT	
OUT	STOP	
OUTPUT	SWITCH	
OWNER		
	T16-6510	
PARAM	T16-6520	
PATHMON	T16-6530	
PATHWAY	T16-6530WP	
POWERONRECOVERY	T16-6540	
PRI	THAW	
PRIMARY	TIMEOUT	

Pathway/iTS Objects

Reserved words exclusive to the HP NonStop PathwayiTS (requester-oriented) objects appear in the following table:

ABORT

BREAK

PRINTER

CHECK-DIRECTORY

REFRESH-CODE

CODEAREALEN

RESUME

CURRENT

RUN

DIAGNOSTIC

SERVERPOOL

DISPLAY-PAGES

STATE

SUSPEND

ECHO

SWAP

ERROR-ABORT

TCLPROG

INITIAL

TCP

INSPECT

TELL

IOPROTOCOL

TERM

IS-ATTACHED

TERMBUFF

TERMPPOOL

MAXINPUTMSGLEN

TRAILINGBLANKS

MAXINPUTMSG

TYPE

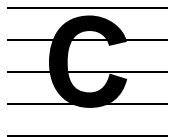
MAXPATHWAYS

MAXREPLY

MAXSERVERPROCESSES

MAXTERMDATA

NONSTOP



Configuration Limits and Defaults

[Table C-1](#) lists global limits for various items within each PATHMON environment.

Note. These limits are subject to change with product SPRs or with new software releases.

Table C-1. Global PATHMON Environment Limits

Item	Limits Per PATHMON Environment
PROGRAM objects	Maximum of 4095 supported by each PATHMON process; suggested maximum of 4094 to avoid error 1108. Maximum of 100 concurrent waited RUN PROGRAM requests from PATHCOM processes and START PROG requests from SPI processes; maximum of 4095 if using nowaited operations.
TCP objects	Maximum of 15 different POBJ files per TCP. Using the HP <i>Inspect</i> symbolic debugger: Each TCP can control a maximum of 8 terminals simultaneously running the Inspect utility, and can process a total of 20 program breakpoints for all SCREEN COBOL programs under inspection. Maximum concurrent links per TCP to the same server process is the value of LINKDEPTH.
TERM objects	Maximum of 4095 supported by each PATHMON process; suggested maximum of 4094 to avoid error 1101.

[Table C-2](#) on page C-2 lists each PATHCOM parameter that has a value range or limit, its corresponding SPI token or field name, its value range, and its default value. This table is organized alphabetically, by PATHCOM parameter. These limits and defaults are subject to change with product SPRs or with new software releases.

SPI tokens and fields are used in the management programming interface to PATHMON environments. For more information about this interface, see the *TS/MP Management Programming Manual*.

Table C-2. Limits and Defaults for Parameters (page 1 of 6)

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults			
AUTORESTART	ZAUTORESTART	Values:	0 through 32,767 Value should be greater than 0.		
		Default:	0		
BACKUPCPU	ZCPU ZCPUPAIR	Values:	0 through 15 Value should be a CPU that is not physically adjacent to the primary CPU.		
		Default:	Depends on command		
BREAK	ZBREAK	Values:	ON OFF		
		Default:	OFF		
CHECK-DIRECTORY	ZCHECKDIRECTORY	Values:	ON OFF Suggested value is OFF for production, ON for development.		
		Default:	ON		
CODEAREALEN	ZCODEAREALEN	Values:	0 through 2,147,483,647 bytes Use SCUP (SCREEN COBOL Utility Program) or STATS TCP to determine this value.		
		Default:	65,536		
COUNT	N.A.	Values:	0 through 65,536		
		Default:	1		
DEBUG	ZDEBUG	Values:	ON OFF		
		Default:	OFF		
DIAGNOSTIC	ZDIAGNOSTIC	Values:	ON OFF		
		Default:	ON		
DISPLAY-PAGES	ZDISPLAYPAGES	Terminal Type	Minimum	Maximum	Default
		T16-6540	1	16 (1)	7
		T16-6530WP	1	8 (1)	7
		T16-6530	1 (2)	8 (1)	7
		T16-6520	1 (2)	3	3
		IBM-3270	1	1	1
		TS530	1	8 (1)	8
Note (1): This is the absolute maximum allowed in the TCP. However, if the terminal or terminal emulator has less memory, the working value is negotiated downward.					
Note (2): Although the PATHMON process allows you to set the number of pages to 1, some terminals or terminal emulators do not allow that setting. For example, the T16-6520 terminal always sets its number of display pages to 3, regardless of the user setting.					
* PATHCOM is the interactive management interface.					
** SPI is the DSM management programming interface.					

Table C-2. Limits and Defaults for Parameters (page 2 of 6)

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
DUMP	ZDUMP	Values: ON OFF Suggested value is ON. Default: OFF
ECHO	ZECHO	Values: ON OFF CURRENT Default: ON
ERROR-ABORT	ZERRORABORT	Values: ON OFF Default: ON
EXCLUSIVE	ZEXCLUSIVE	Values: ON OFF Default: OFF
FILE	ZFILE	Values: One through seven alphanumeric characters, including the \$; the first character after the \$ must be alphabetic. Default: N.A.
HOMETERM	ZHOMETERM	Values: Name of home terminal for servers in class. Default: Home terminal of PATHMON.
INSPECT	ZINSPECT	Values: ON OFF Suggested value is OFF for production. Default: OFF
IOPROTOCOL	ZIOPROTOCOL	Values: 0 1 Default: 0
IS-ATTACHED	ZISATTACHED	Values: YES NO or T16-6530 T16-6520 IBM-3270 The value depends upon the command in which the parameter is used. Default: N.A.
MAXINPUTMSGLEN	ZMAXINPUTMSGLEN	Values: 0 through 6000 bytes; does not include message header length. Default: 133
MAXINPUTMSG	ZMAXINPUTMSG	Values: 0 through 2045 Default: 0
MAXPATHWAYS	ZMAXPATHWAYS	Values: 0 through 4095 Default: 0
MAXREPLY	ZMAXREPLY	Values: 0 through 32,767 bytes Default: 2000
* PATHCOM is the interactive management interface.		
** SPI is the DSM management programming interface.		

Table C-2. Limits and Defaults for Parameters (page 3 of 6)

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
MAXSERVERCLASSES	ZMAXSERVERCLASSES	Values: 0 through 4095 Total of MAXSERVERS values for all server classes must not exceed value of MAXSERVERPROCESSES. Default: N.A.
MAXSERVERPROCESS ES	ZMAXSERVERPROCESS ES	Values: 0 through 4095 Default: N.A.
MAXTCPS	ZMAXTCPS	Described in <i>TS/MP Management Programming Manual</i> .
MAXTELLQUEUE	ZMAXTELLQUEUE	Described in <i>TS/MP Management Programming Manual</i> .
MAXTELLS	ZMAXTELLS	Described in <i>TS/MP Management Programming Manual</i> .
MAXTERMDATA	ZMAXTERMDATA	Values: If NONSTOP set to 0: 0 through 2,147,483,647 bytes. If NONSTOP set to 1: 2804 through 2,147,483,647 bytes. Default: 8000
MAXTERMS	ZMAXTERMS	Values: 0 through 4095 Default: N.A.
NONSTOP	ZNONSTOP	TCPs: Values: 0 1 Default: 1
POWERONRECOVERY	ZPOWERONRECOVERY	Values: ON OFF Default: ON
PRI	ZPRIORITY	Values: 1 through 199 Default (PATHMON): The priority assigned by \$CMON, or if \$CMON is not active, the priority of TACL minus 1. Default (TCP): 20 less than the priority of PATHMON. Default (SERVER): 10 less than the priority of PATHMON.
PROCESS	ZLINKMON ZPROCESS	Values: One to six alphanumeric characters, including the \$; first character after the \$ must be alphabetic. Default: N/A

* PATHCOM is the interactive management interface.

** SPI is the DSM management programming interface.

Table C-2. Limits and Defaults for Parameters (page 4 of 6)

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
PROGRAM	ZPROGRAM	<p>Values: For <i>Guardian</i> operating environment server processes, the name of the Guardian object file name; 1 through 15 alphanumeric or hyphen characters; first character must be alphabetic.</p> <p>Default: None</p>
SECURITY	ZSECURITY	<p>Values: "A" Any local user "G" A group member or owner "O" Owner only "." Local super ID "N" Any local or remote user "C" Any member of owner's community "U" Any member of owner's user class</p> <p>The owner should not be a super-group user with "N" or "A" for a security value; this combination causes a security breach.</p> <p>Default: "U"</p>
SERVERPOOL	ZSERVERPOOL	<p>Values: 32 through 2,147,483,647 bytes</p> <p>Use the following formula to determine the maximum value of SERVERPOOL:</p> <p>$SERVERPOOL = MAXREPLY * (\text{the minimum number of servers the TCP can connect to, or the number of terminals that the TCP controls})$</p> <p>The space allocated for SERVERPOOL is the specified value rounded up to the next-highest multiple of 4.</p> <p>The value for this parameter is typically in the range from 10,000 to 30,000 bytes.</p> <p>Default: 20,000</p>
STATS	ZSTATS	<p>Values: ON OFF</p> <p>Default: OFF</p>
TCLPROG	ZTCLPROG	<p>Values: A file name consisting of all characters prefixed to the letters COD in the name under which the SCREEN COBOL program is compiled.</p> <p>Default: N.A.</p>
TCP	ZTCP	<p>Values: 1 through 15 alphanumeric or hyphen characters; first character must be alphabetic.</p> <p>Default: N.A.</p>
<p>* PATHCOM is the interactive management interface.</p> <p>** SPI is the DSM management programming interface.</p>		

Table C-2. Limits and Defaults for Parameters (page 5 of 6)

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
TERM	ZTERM	Values: 1 through 15 alphanumeric or hyphen characters; first character must be alphabetic. Default: N.A.
TERMBUF	ZTERMBUF	Values: 256 through value of TERMPOOL If writing diagnostic display messages to terminal: 1315 through value of TERMPOOL. Default: 1500
TERMPOOL	ZTERMPOOL	Values: minimum of 25 Use the following formula to determine the value of TERMPOOL: $\text{TERMPOOL} = \text{MAXTERMS} * (\text{the largest value of either TERMBUF or the largest possible I/O (display, accept, or message)})$ The space allocated for TERMPOOL is the specified value rounded up to the next-highest multiple of 4. Default: 10,000
* PATHCOM is the interactive management interface. ** SPI is the DSM management programming interface.		

Table C-2. Limits and Defaults for Parameters (page 6 of 6)

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
TMF	ZTMF	Values: ON OFF Default (PROGRAM): ON Default (TERM): ON Default (SERVER): OFF
TRAILINGBLANKS	ZTRAILINGBLANKS	Values: ON OFF Default: ON
TYPE	ZTERMTYPE	Values: 1:n or IBM-3270:n 3:0 or T16-6520:0 4:0 or T16-6530:0 5:0 or CONVERSATIONAL:0 6:0 or T16-6540:0 7:n or INTELLIGENT:n 8:0 or T16-6530WP:0 Subtypes for INTELLIGENT devices: 7:0 WRITEREAD I/O protocol (conversational mode) 7:1 WRITEand READ I/O protocol (block mode) 7:2 WRITEREAD I/O protocol (block mode) Default:7:0 Subtypes for IBM-3270 devices: 1:1 IBM 3277 M1 1:2 IBM 3277 M2 1:3 IBM 3278 M3 1:4 IBM 3278 M4 1:5 IBM 3278 M1 1:6 IBM 3278 M5 Default: Device type determined by DEVICEINFO query.

* PATHCOM is the interactive management interface.

** SPI is the DSM management programming interface.

D Migration Information

This appendix covers migration and compatibility issues for Pathway applications that include the HP NonStop Pathway/iTS product on C-series and D-series systems. The following topics are discussed:

- Interprocess communication issues
- Application conversion

For more information about running applications on systems running the D-series RVU, see the *Guardian Application Conversion Guide*.

Note. There is no direct migration path for PATHMON environments from C-series systems to D40.00 or later systems running the D-series RVU. In addition, interoperability between PATHMON environment processes on C-series systems and D40 or later systems running the D-series RVU is not supported.

For information about migrating a PATHMON environment from a C-series system to an earlier-release D-series system, and also for information about interoperation between PATHMON environment processes on C-series and systems running the D-series RVU, see the “Migration Information” appendix in the D30.02 version of the *TS/MP and Pathway System Management Guide*.

Interprocess Communication Issues

The Pathway/iTS product supports D-series operating system features as follows:

- Requester processes can run at high and low PINs.
- Terminal control processes (TCPs) can run at a high PIN under the following conditions:
 - The Pathway/iTS HIGHPIN option is set to ON for the TCP object. (The default value is OFF.)
 - The ?HIGHPIN option is set by compiler or binder directive in the object file.
 - There is a high PIN available.

Alternatively, to force a TCP to run at a low PIN, even if the other conditions for running at a high PIN are satisfied, leave the HIGHPIN option set to OFF (the default value) or, to document your intention, set the HIGHPIN option to OFF with the SET TCP command.

If you have configured your PATHMON environment to allow the use of high PINs, you can define additional server classes without exhausting PINs that you might need for other processes. Additionally, you have more freedom to define servers as static, thus reducing the overhead associated with process startup.

D-series processes and associated application processes can run in many combinations of high and low PINs. [Table D-1](#) lists the possible communication paths for D-series Pathway/iTS processes.

If a process runs at a high PIN, all processes that communicate with that process must be able to communicate with a high-PIN process. The characters "N.A." indicate that an option is not available.

Table D-1. Pathway/iTS Process High-PIN and Low-PIN Support

Operation	PATHTCP2	SCREEN COBOL	SCUP
Run at a high PIN	Yes	No	No
Service high-PIN server	Yes	N.A.	N.A.
Service high-PIN requesters	Yes	N.A.	N.A.
Open a high-PIN process	Yes	Yes	Yes
Create a high-PIN process	N.A.	N.A.	N.A.
Can be created by a high-PIN process	Yes	Yes	Yes

TCPs can be created at high PINs if the following conditions are met:

- The HIGHPIN option is set to ON for the TCP object. (The default value is OFF.)
- The HIGHPIN option is enabled in the object file.
- There is a high PIN available.

Application Conversion Guidelines

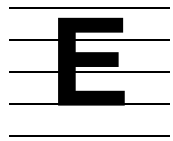
Pathway applications originally developed for use on C-series systems can generally run at low PINs without code change or recompilation. You might need to make the following change in your initialization files, however: if you use a default subvolume name in a file name that contains the volume name, include a subvolume specification in the file name.

To run at a high PIN, a Pathway requester or server program must call selected D-series operating system procedures and use HIGHPIN compile and bind options that allow the program to run at a high PIN.

Additional considerations specific to the PATHMON environment include the following:

- Process names for TCPs must follow the C-series representation of a dollar sign (\$) followed by one to four alphanumeric characters.
- Device names (for terminals, printers, tape drives, and so on) must follow the C-series representation: a dollar sign (\$) followed by one to six alphanumeric characters.
- A process must run named if it runs at a high PIN and opens the PATHMON process. You can use the ?RUNNAMED compiler directive to ensure that a process runs named.

For more information about converting your applications, see the *Guardian Application Conversion Guide*.



Setting TMF Parameters

When you are configuring and controlling a Pathway application that uses TMF, consider the following three basic questions:

- How do the settings you specify for the TMF parameter of the SET TERM and SET PROGRAM commands affect SCREEN COBOL SEND statements?
- How is the TCP checkpointing strategy affected by the settings you specify for the TMF parameter of the SET SERVER command? (For more information about the SET SERVER command beyond that presented in this appendix, see the *TS/MP System Management Manual*).
- What problems are caused by using the TMF OFF option of the SET TERM or SET PROGRAM commands as a switch to turn TMF off for a requester that is communicating with servers running under the TMF software?

Addressing these questions helps ensure the consistency of the database and helps you to improve the reliability and performance of the applications that use the database.

SET TERM and SET PROGRAM Commands and TMF

The SET TERM and SET PROGRAM commands each contain a TMF parameter with an ON or OFF option:

- TMF ON—The TCP invokes the corresponding operating system procedure for any TMF statement issued from a SCREEN COBOL program. ON is the default setting whether or not the TMF software is running.
- TMF OFF—The TCP does not invoke the corresponding operating system procedure for any TMF statement issued from a SCREEN COBOL program. Instead, the TMF statement appears (to the SCREEN COBOL program) to complete successfully and the program can continue to execute.

For most Pathway applications, you should use the default parameter settings, whether or not the TMF software is running.

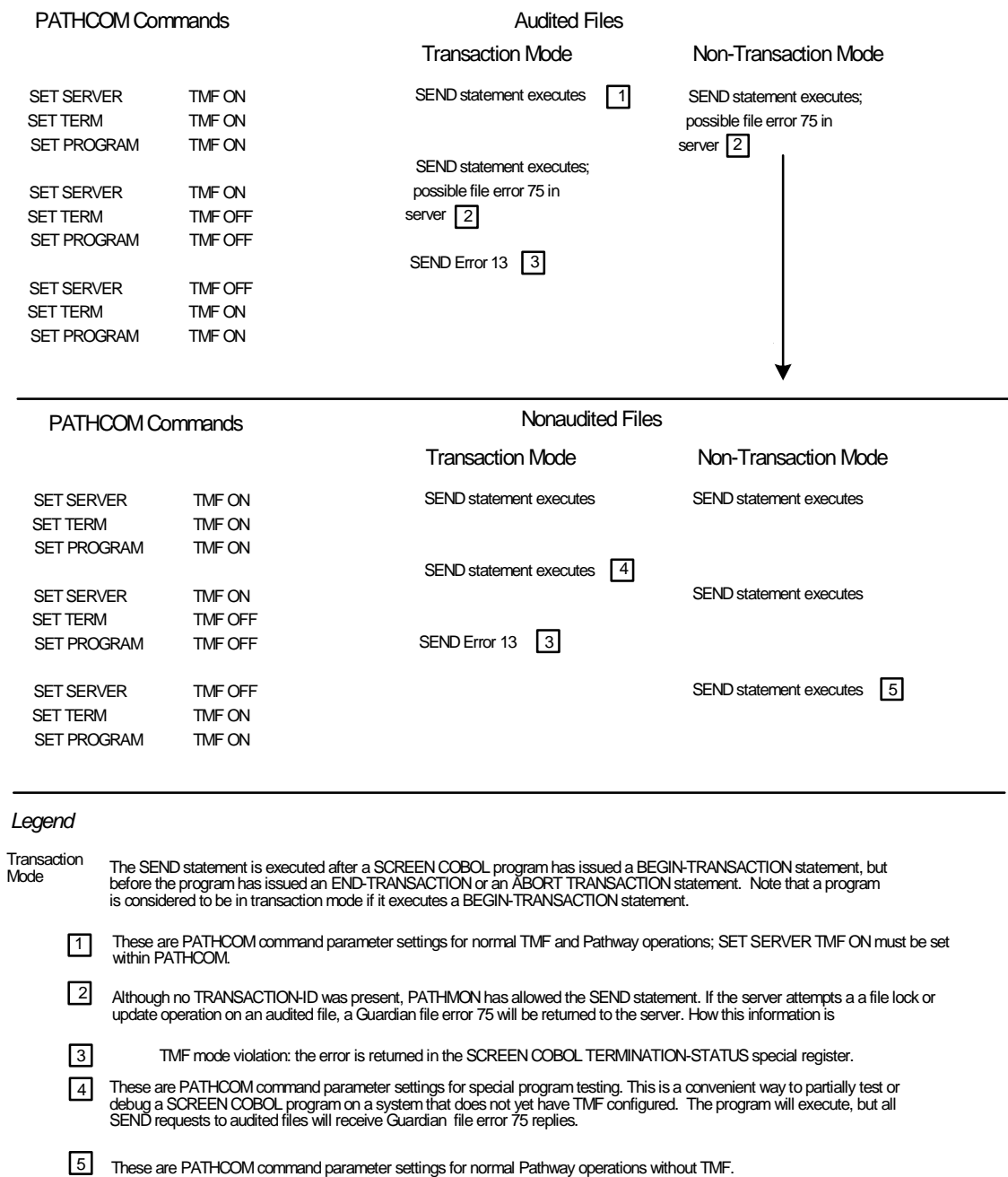
Effect of TMF Parameters in PATHCOM on SCREEN COBOL SEND Operations

[Figure E-1](#) on page E-3 lists how the various combinations of settings of the TMF parameter in the PATHCOM SET SERVER, SET TERM, and SET PROGRAM

commands affect a SCREEN COBOL SEND statement when the PATHMON process and the TMF subsystem are both running on the system.

Note. The SET SERVER command is used to set attributes for SERVER objects created and controlled under TS/MP. For more information about SERVER objects and the server environment, see the *TS/MP System Management Manual* and the *TS/MP Management Programming Manual*.

Figure E-1. SEND Operations With TMF



VST035.vsd

Depending on the type of file access attempted, the TCP either allows the SEND statement to execute or issues the appropriate error message.

In a PATHMON environment that normally runs with the TMF subsystem, do not use the following commands to turn off TMF subsystem operations temporarily:

```
SET TERM TMF OFF  
SET PROGRAM TMF OFF
```

The condition resulting from these commands appears to allow normal operation, because the BEGIN-TRANSACTION statement that would have failed if the TMF subsystem were stopped now appears to work; the TCP allows a SEND operation to a server that can access and update only nonaudited files. Files updated by servers are not protected by the TMF subsystem, and the TCP does not perform checkpoints before or after SEND statements.

Timeouts on SEND Operations to Servers

Although the syntax of the SCREEN COBOL SEND statement does not include a TIMEOUT clause, you can effectively supply one with the PATHCOM SET SERVER command. When you include a TIMEOUT clause in the SET SERVER command, all SENDs to that server class are timed by the TCP. If the specified number of seconds elapses after a SEND operation is initiated and before a reply is received, the TCP issues a *Guardian* operating environment CANCEL procedure call against the outstanding I/O to the server. If the SEND operation was performed while the requester program was in transaction mode, the transaction is automatically aborted by the file-system. In such a case, the requester program discovers that the transaction was aborted when it subsequently attempts to update the database (with another SEND statement) or issues an END-TRANSACTION statement.

TCP Checkpointing Strategy

For PATHMON environments with the TMF subsystem running, the TCP uses the following checkpointing strategy:

- At the BEGIN-TRANSACTION statement, a full copy of the task's context is made to a secondary area (slot 1) in the extended data segment, and a checkpoint to the backup is performed.
- At the END-TRANSACTION statement, a full-context checkpoint is performed.
- At the SEND statement with the SET SERVER TMF parameter defined as OFF, a checkpoint is performed before and after the SEND statement when the SCREEN COBOL program is outside of transaction mode. (Any time a SEND operation is performed outside of a transaction boundary and the server attempts to lock or update a record in an audited file, the operation fails with a Guardian error 75.)
- At the SEND statement with the SET SERVER TMF parameter defined as ON, no checkpoints are performed, whether or not the SCREEN COBOL program is in transaction mode. This means that SEND requests to TMF protected servers that

operate on audited files require fewer checkpoints than SEND requests to servers that do not operate under TMF protection.

TCP checkpointing requirements can be significantly reduced if Pathway applications running with the TMF subsystem have TMF protected servers read outside of transaction mode before updating the database.

-
- △ **Caution.** If a SEND request outside of transaction mode is sent to a TMF protected server that operates on nonaudited files, data might be lost because the TMF subsystem is not invoked and the TCP performs fewer checkpoints.
-

You can improve the performance of a Pathway application by taking advantage of the TCP checkpointing strategy for TMF protected servers, as follows:

- Do not use transaction mode for a server with read-only access to a database if the requester displays the data before any attempt is made to change the data. In the event of a failure, the read operations are retryable and fault-tolerant operation is maintained.
- Do not use transaction mode for a server that writes to an entry-sequenced logging file in which duplicates are acceptable. In the event of a failure, the requester can retry the write operations, so there is no need to back out the write. In contrast, a key-sequenced file requires a backout; otherwise, the transaction will fail when the second write is attempted at the same location.

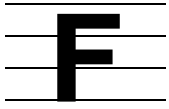
Note. SERVER objects are created and controlled under TS/MP. For more information about SERVER objects and the server environment, see the *TS/MP System Management Manual* and the *TS/MP Management Programming Manual*.

Precautions for Using TMF Parameters

If a TMF error occurs and makes normal operations impossible, setting the TMF parameter options to OFF is not the solution for continuing normal operations. Setting these options can have the following result:

- A SCREEN COBOL program that uses ABORT-TRANSACTION or RESTART-TRANSACTION to handle exceptions to normal program operation only appears to execute; the TMF verbs have no effect.
- With the SET SERVER TMF parameter defined as ON and the SET TERM TMF or SET PROGRAM TMF parameters defined as OFF, the TCP makes checkpoint, retry, and syncdepth decisions as if the TMF software were running. For example, the TCP performs fewer checkpoints and opens servers with a sync depth of 0 instead of 1. In this case, the TCP is not running in fault-tolerant mode, and a single CPU failure can cause the application to fail.

To determine how to address the TMF error, see the *TMF Operations and Recovery Guide*.



Setting the DISPLAY-PAGES Parameter

This appendix explains how to calculate the DISPLAY-PAGES parameter of the SET TERM and SET PROGRAM commands.

Note. For more information on the SET TERM and SET PROGRAM commands, including syntax descriptions, see [Section 10, TERM Commands](#), and [Section 11, PROGRAM Commands](#).

Screen Caching

The DISPLAY-PAGES parameter specifies the depth of the terminal's screen caching. The 6520, 6530, and 6540 terminals and terminal emulators have screen caching capability to increase performance. Performance is improved because screen caching prevents a screen from being built a second time after the initial building of the screen.

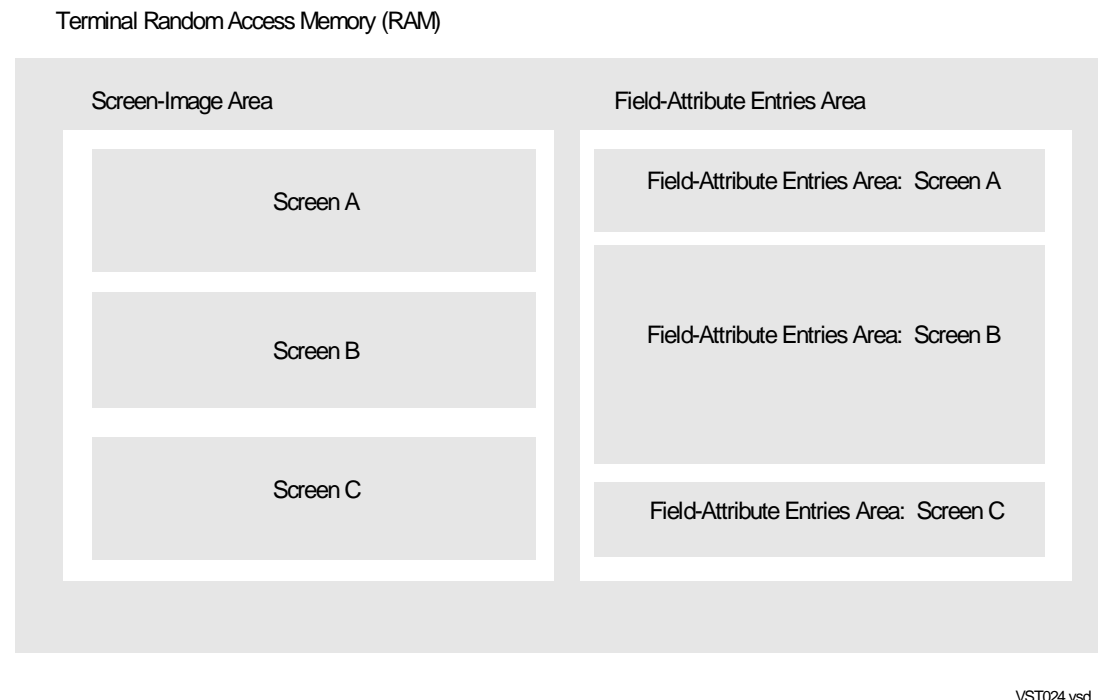
For example, when the first DISPLAY BASE operation for Screen A is executed, all the terminal escape sequences needed to build the screen are written to the terminal. The terminal stores the screen image and field definitions in its memory. When the next DISPLAY BASE operation for Screen A is done, only a few escape sequences—enough to display the stored screen—are written to the terminal. Such screen caching results in a significant savings in terminal process time.

Terminal Memory Organization

The value of the DISPLAY-PAGES parameter affects how the terminal's fixed amount of RAM (random access memory) is used. RAM is divided into two separate areas: the screen-image area and the field-attribute entries area. There is one screen-image area associated with a screen definition and approximately two field-attribute entries associated with each field definition. A field-attribute entry is an entry in a table that the terminal maintains to record field characteristics (for example, brightness or blinking versus non-blinking).

The DISPLAY-PAGES parameter determines how much terminal memory is used for the screen-image area compared to how much is used for the field-attribute entries area. The more memory assigned to one, the less available to the other.

[Figure F-1](#) on page F-2 illustrates a terminal's RAM organization with a DISPLAY-PAGES setting of 3. The RAM space reserved in the screen-image area stores the image of Screen A, Screen B, and Screen C. The remainder of the RAM is used to store the field-attribute entries for each screen field.

Figure F-1. RAM Organization Within a Terminal

Determining the DISPLAY-PAGES Value

The correct value of the DISPLAY-PAGES parameter depends on the application being run. An inappropriate value for DISPLAY-PAGES can result in the following consequences:

- If the value of DISPLAY-PAGES is too large, there will not be enough room in the terminal's field-attribute area for all the field-attribute entries. This causes the terminal to abort with the terminal error ALL FIELDS USED.
- If the value of DISPLAY-PAGES is too small, the terminal's RAM will not be used to its full capacity. Consequently, performance is not as good as it could be because the number of screens that could be recalled is not at its maximum.

The factors that determine the correct value for the DISPLAY-PAGES parameter are the number of field-attribute entries per screen definition, and terminal capacity. The information in the following paragraph helps you estimate the number of field-attribute entries. The tables in the remainder of this section help you match terminal type to the average number of field-attribute entries to determine the appropriate DISPLAY-PAGES value. Using the appropriate DISPLAY-PAGES value allows a terminal to achieve its highest performance level without exceeding its fixed amount of RAM.

Estimating Field-Attribute Entries

Although the actual number of field-attribute entries a screen definition contains depends on the layout of the screen, you can estimate the number of field-attribute entries by multiplying the number of screen fields by 2. For example, a screen with 10 input fields has approximately 20 field-attribute entries. The number of field-attribute entries is twice the number of screen fields because the terminal also identifies the spaces between the screen fields as fields.

Assessing Terminal Capacity

“Terminal capacity” refers to the number of field-attribute entries that a terminal can store given a specific DISPLAY-PAGES setting. The following tables list the average number of field-attribute entries that individual terminals can store for each value of DISPLAY-PAGES.

[Table F-1](#) lists the average number of field-attribute entries a 6520 terminal can store. Note that the number of pages that a 6520 terminal can cache is fixed at 3.

Table F-1. Storage Capacity of 6520 Terminal

DISPLAY-PAGES Value	Average Number of Field-Attribute Entries per Screen
3	1920

[Table F-2](#) lists the average number of field-attribute entries a 6526 terminal can store.

Table F-2. Storage Capacity of 6526 Terminal

DISPLAY-PAGES Value	Average Number of Field-Attribute Entries per Screen
1	3265
2	1524
3	944
4	654
5	480
6	364
7	281
8	219

[Table F-3](#) on page F-4 lists the average number of field-attribute entries a 6530 terminal can store.

Table F-3. Storage Capacity of 6530 Terminal

DISPLAY-PAGES Value	Average Number of Field-Attribute Entries per Screen
1	4095
2	2510
3	1488
4	976
5	670
6	465
7	319
8	209

[Table F-4](#) lists the average number of field-attribute entries a TS530 terminal can store. Note that the number of pages that the TS530 terminal can cache is fixed at 8.

Table F-4. Storage Capacity of TS530 Terminal

DISPLAY-PAGES Value	Average Number of Field-Attribute Entries per Screen
8	1980

Source Code for Programs in Section 6

This appendix presents the source code for the screen and server programs used in the examples described in [Section 6, Examples of System Management Tasks](#).

You can configure and run a real PATHMON environment based on this example. To create this environment:

1. Enter the source code for the screen program into a text file named SC (using your text editor).
2. Enter the source code for the server program into a text file named SV (using your text editor).
3. Follow the directions in [Section 6, Examples of System Management Tasks](#), to compile the source code into object programs, configure and start your environment, configure and start the individual objects for that environment, and run the application.

Source Code for Screen Program

The source code for the screen program (to be entered in the text file named SC) appears in [Example G-1](#) on page G-2. This code is written in the SCREEN COBOL language.

This program is intended for use on user terminals running in block mode. However, screen programs can also be written to handle terminals operating in conversational mode.

Example G-1. Source Code for the Screen Program (page 1 of 3)

IDENTIFICATION DIVISION.

PROGRAM-ID. EXAMPLE-SCREEN-PROGRAM.

ENVIRONMENT DIVISION.

CONFIGURATION SECTION.

SOURCE-COMPUTER. T16.

OBJECT-COMPUTER. T16,
TERMINAL IS T16-6520.

SPECIAL-NAMES.

F1-KEY IS F1, F2-KEY IS F2, F15-KEY IS F15, F16-KEY IS F16
SF1-KEY IS SF1, SF16-KEY IS SF16, ATTENTION IS REVERSE.

DATA DIVISION.

WORKING-STORAGE SECTION.

01 WS.

02 ERROR-MSG PIC X(64).

01 EXIT-FLAG PIC 9 VALUE 0.

88 EXIT-PROGRAM VALUE 1.

01 SEND-ERROR-FLAG PIC 99.

01 ENTRY-MSG.

02 PW-HEADER.

04 REPLY-CODE PIC S9(4) COMP.

04 FUNCTION-CODE PIC XX.

02 MESSAGE-NUMBER PIC 9.

01 ENTRY-REPLY.

02 PW-HEADER.

04 REPLY-CODE PIC S9(4) COMP.

04 FILLER PIC X(2).

02 SERVER-RECORD PIC X(64).

01 ERROR-REPLY.

02 REPLY-CODE PIC 9(4) COMP.

02 FILLER PIC X(2).

02 ERROR-CODE PIC 999 COMP.

Example G-1. Source Code for the Screen Program (page 2 of 3)

SCREEN SECTION.

```

01 EXAMPLE-SCREEN BASE SIZE 24, 80.
  03 FILLER          AT 1, 20  VALUE "EXAMPLE SCREEN COBOL
                                PROGRAM".
  03 FILLER          AT 10, 1  VALUE "MESSAGE NUMBER :".
  03 MESSAGE-NUMBER AT 10, * + 2 PIC 9, USING
                                MESSAGE-NUMBER OF ENTRY-MSG.
  03 MESSAGE-TEXT   AT 12, 1 PIC X(64)
                                FROM SERVER-RECORD OF ENTRY-REPLY.
  03 FILLER          AT 21, 15 VALUE
                                "F1 - READ      F16 - EXIT PROGRAM".
  03 ERROR-MSG      AT 24, 2 PIC X(64) ADVISORY FROM
                                ERROR-MSG OF WS.

```

PROCEDURE DIVISION.

A-MAIN.

```

  DISPLAY BASE EXAMPLE-SCREEN.
  MOVE 0 TO MESSAGE-NUMBER OF ENTRY-MSG.
  DISPLAY MESSAGE-NUMBER OF EXAMPLE-SCREEN.
  PERFORM CASE-MANAGER UNTIL EXIT-PROGRAM.

```

A-EXIT.

```

  EXIT PROGRAM.

```

CASE-MANAGER.

```

  ACCEPT MESSAGE-NUMBER OF EXAMPLE-SCREEN UNTIL F1-KEY
  ESCAPE ON F16-KEY (F2-KEY THROUGH F15-KEY, SF1-KEY
  THROUGH SF16-KEY).
  PERFORM ONE OF
    READ-MESSAGE, SET-EXIT, KEY-NOT-SUPPORTED
  DEPENDING ON TERMINATION-STATUS.

```

READ-MESSAGE.

```

  MOVE "01" TO FUNCTION-CODE OF ENTRY-MSG.
  PERFORM SEND-DATA.

```

SET-EXIT.

```

  MOVE 1 TO EXIT-FLAG.

```

KEY-NOT-SUPPORTED.

```

  MOVE "FUNCTION KEY NOT SUPPORTED" TO ERROR-MSG OF WS.
  PERFORM DISPLAY-ADVISORY.

```

Example G-1. Source Code for the Screen Program (page 3 of 3)

```

SEND-DATA.
  MOVE 0 TO SEND-ERROR-FLAG.
  SEND ENTRY-MSG TO "EXAMPLE-SERVER"
    REPLY CODE 0      YIELDS ENTRY-REPLY
    CODE 999  YIELDS ERROR-REPLY ON ERROR PERFORM
                                     SEND-ERROR.

  IF SEND-ERROR-FLAG = 99
    DISPLAY TEMP "SEND ERROR" IN
      ERROR-MSG OF EXAMPLE-SCREEN
    ELSE PERFORM ONE OF PARA-1
      PARA-2
    DEPENDING ON TERMINATION-STATUS.
  PARA-1.
    MOVE SPACES TO ERROR-MSG OF WS
    DISPLAY MESSAGE-TEXT OF EXAMPLE-SCREEN,
      ERROR-MSG OF EXAMPLE-SCREEN.
  PARA-2.
    PERFORM SETUP-SERVER-ERROR
    PERFORM DISPLAY-ADVISORY.

SEND-ERROR.
  MOVE 99 TO SEND-ERROR-FLAG.

SETUP-SERVER-ERROR.
  IF ERROR-CODE = 1
    MOVE "SERVER FUNCTION NOT SUPPORTED" TO ERROR-MSG
      OF WS
  ELSE
    IF ERROR-CODE = 2
      MOVE "MESSAGE DOES NOT EXIST FOR SPECIFIED NUMBER"
        TO ERROR-MSG OF WS
    ELSE
      MOVE "UNKNOWN SERVER ERROR" TO ERROR-MSG OF WS.

DISPLAY-ADVISORY.
  DISPLAY TEMP ERROR-MSG OF EXAMPLE-SCREEN.
  TURN TEMP ATTENTION IN ERROR-MSG OF EXAMPLE-SCREEN.

```

Source Code for Server Program

The source code for the server program (to be entered in the text file named SV) appears in [Example G-2](#). This code is written in COBOL and can be compiled using the COBOL85 compiler.

Example G-2. Source Code for the Server Program (page 1 of 2)

```
IDENTIFICATION DIVISION.
    PROGRAM-ID. EXAMPLE-SERVER.

ENVIRONMENT DIVISION.
    CONFIGURATION SECTION.
    SOURCE-COMPUTER. TANDEM/16.
    OBJECT-COMPUTER. TANDEM/16.
    INPUT-OUTPUT SECTION.
        FILE-CONTROL.
            SELECT MESSAGE-IN, ASSIGN TO $RECEIVE
            FILE STATUS IS RECEIVE-FILE-STATUS.
            SELECT MESSAGE-OUT, ASSIGN TO $RECEIVE
            FILE STATUS IS RECEIVE-FILE-STATUS.
        RECEIVE-CONTROL.
            TABLE OCCURS 5 TIMES.

DATA DIVISION.
    FILE SECTION.
    FD MESSAGE-IN
        LABEL RECORDS ARE OMITTED.
    01 ENTRY-MSG.
        02 PW-HEADER.
            04 REPLY-CODE          PIC S9(4) COMP.
            04 FUNCTION-CODE      PIC XX.
            02 MESSAGE-NUMBER     PIC 9.

    FD MESSAGE-OUT
        LABEL RECORDS ARE OMITTED
        RECORD CONTAINS 1 TO 68 CHARACTERS.
    01 ENTRY-REPLY.
        02 PW-HEADER.
            04 REPLY-CODE          PIC S9(4) COMP.
            04 FILLER              PIC X(2) .
            02 SERVER-RECORD       PIC X(64) .
    01 ERROR-REPLY.
        02 REPLY-CODE             PIC S9(4) COMP.
        02 FILLER                 PIC X(2) .
        02 ERROR-CODE             PIC 999 COMP.
```

Example G-2. Source Code for the Server Program (page 2 of 2)

```

WORKING-STORAGE SECTION.
  01 RECEIVE-FILE-STATUS.
    02 STAT-1          PIC 9.
      88 CLOSE-FROM-REQUESTOR VALUE 1.
    02 STAT-2          PIC 9.
  01 WS.
    02 ERROR-CODE      PIC 999  COMP.

PROCEDURE DIVISION.
BEGIN-COBOL-SERVER.
  OPEN INPUT MESSAGE-IN.
  OPEN OUTPUT MESSAGE-OUT SYNCDEPTH 1.
  PERFORM B-TRANS UNTIL CLOSE-FROM-REQUESTOR.
  STOP RUN.

B-TRANS.
  READ MESSAGE-IN, AT END STOP RUN.
  MOVE PW-HEADER OF MESSAGE-IN TO PW-HEADER OF
                                     MESSAGE-OUT.

  MOVE 0 TO ERROR-CODE OF WS
  IF FUNCTION-CODE = "01"
    PERFORM READ-MESSAGE
  ELSE
    PERFORM FUNCTION-NOT-SUPPORTED.

  IF ERROR-CODE OF WS = 0
    MOVE 0 TO REPLY-CODE OF ENTRY-REPLY
    WRITE ENTRY-REPLY
  ELSE
    MOVE 999 TO REPLY-CODE OF ERROR-REPLY
    MOVE ERROR-CODE OF WS TO ERROR-CODE OF ERROR-REPLY
    WRITE ERROR-REPLY.

READ-MESSAGE.
  IF MESSAGE-NUMBER = 0
    MOVE "THIS IS MESSAGE ZERO" TO SERVER-RECORD
  ELSE IF MESSAGE-NUMBER = 1
    MOVE "THIS IS MESSAGE NUMBER ONE" TO SERVER-RECORD
  ELSE IF MESSAGE-NUMBER = 2
    MOVE "THIS IS MESSAGE NUMBER TWO" TO SERVER-RECORD
  ELSE
    PERFORM INVALID-MESSAGE-NUMBER.

FUNCTION-NOT-SUPPORTED.
  MOVE 1 TO ERROR-CODE OF WS.

INVALID-MESSAGE-NUMBER.
  MOVE 2 TO ERROR-CODE OF WS.

```

Index

Numbers

6540 cache error... (message 3057) [13-17](#)

A

ABORT TERM command [3-9](#), [10-2/10-3](#)

ACCEPT display field [9-40](#)

ACCEPT Statement [5-22](#)

ACCEPT statement [10-19](#)

Actual number of parameters...
(message 3005) [13-4](#)

Actual parameter size...
(message 3006) [13-4](#)

ADD command

 PROGRAM [2-12/2-13](#)

 TCP [2-7](#)

 TERM [2-11](#)

ADD PROGRAM command [2-12/2-13](#),
[11-2](#)

ADD TCP command [2-7](#), [9-2](#)

ADD TERM command [10-4](#)

Adding

 PROGRAM objects [2-12](#)

 TCP objects [2-7](#)

 TERM objects [2-11](#), [6-5](#)

ALL FIELDS USED error [10-21](#), [11-17](#)

ALTER command [4-15](#)

ALTER PROGRAM command [11-3/11-4](#)

ALTER TCP command [9-3/9-4](#)

ALTER TERM command [10-5/10-6](#)

Altering

 objects

 See individual objects

 PATHMON environment [4-13](#)

Application defined error...
(message 3124) [13-30](#)

Applications

 configuring in a PATHMON
 environment [2-1](#)

Applications (continued)

 multiple per PATHMON
 environment [2-2](#)

Arithmetic overflow (message 3015) [13-8](#)

Attempt to receive unsolicited...
(message 3176) [13-38](#)

Attempt to receive unsolicited...
(message 3178) [13-39](#)

Attribute values

 configuration limits for [C-1/C-7](#)

 copying [9-25](#), [10-23](#), [11-20](#)

Attributes

 See also Attribute values

 for global PATHMON environment
 limits [2-4](#)

 for PROGRAM objects [2-13](#)

 for TCP objects [2-7](#)

 for TERM objects [2-11](#)

AUTORESTART attribute

 SET TCP command [9-19](#)

 SET TERM command [10-19](#)

B

Backup CPU

 for PATHMON [4-15](#)

 for TCP [4-15](#)

 switching with primary [4-16](#)

Backup failed (message 3212) [13-45](#)

Backup process creation failure...
(message 3211) [13-45](#)

Backup process not up (message
1070) [9-14](#), [9-45](#)

Backup processes TCP objects [9-27](#)

Backup processor down (message
1093) [9-8](#)

Backup processor down
(message 3210) [13-45](#)

Backup started (message 3244) [13-55](#)

Backup task error... (message 3070) [13-22](#)

BACKUPCPU attribute CONTROL TCP
 command [9-5](#)
 Because [1-12](#)
 BEGIN-TRANSACTION statement [E-4](#)
 Binder program [2-10](#), [9-23](#)
 BKPT-HOLD in WAIT display field [9-40](#)
 Blank characters, trailing [10-25](#), [11-20](#)
 Block mode terminals [6-1](#), [11-23](#)
 BREAK attribute
 SET PROGRAM command [11-16](#)
 SET TERM command [10-19](#)
 BREAK key function
 configuring for PROGRAM
 objects [2-13](#)
 configuring for TERM objects [2-12](#)
 STOP TCP command [9-42](#)
 Break key function
 PATHCOM commands [7-6](#)
 Buffer space determining size of [5-9](#)

C

Cache error... (message 3056) [13-16](#)
 Cache error... (message 3057) [13-17](#)
 Caching, screen [F-1](#)
 Called program unit not found
 (message 3020) [13-9](#)
 CALL, actual number of...
 (message 3005) [13-4](#)
 CALL, actual parameter size...
 (message 3006) [13-4](#)
 CALL, PROGRAM unit name invalid
 (message 3044) [13-14](#)
 CANCEL procedure, Guardian [10-23](#),
[11-19](#)
 Cancelling PATHCOM commands [7-6](#)
 Cannot call PU with...
 (message 3036) [13-13](#)
 Characters blank [10-25](#), [11-20](#)
 CHAR-SET display field [10-37](#)
 CHECKPOINT in WAIT display field [9-40](#)

Checkpointing
 and TMF [E-4](#)
 as TCP task [5-6](#)
 improving performance with [E-5](#)
 queues maintained for [5-6](#)
 CHECK-DIRECTORY attribute [4-21](#), [9-19](#)
 Client/server computing [1-15](#)
 Code area [5-5](#), [5-12/5-13](#)
 Code area too small...
 (message 3235) [13-52](#)
 Code of received message...
 (message 3164) [13-34](#)
 CODEAREALEN attribute [9-20](#)
 CODEAREALEN attribute, for TCPs [2-7](#)
 COLD start option in command files [6-3](#),
[6-9](#)
 Command files
 examples of [6-2](#), [6-3](#), [6-6](#), [6-8](#), [6-9](#)
 using [7-6](#)
 Commands
 cancelling commands [7-6](#)
 command list [7-2/7-4](#)
 controlling Pathway [8-1/8-4](#)
 controlling PROGRAM
 objects [11-1/11-25](#)
 controlling TCP objects [9-1/9-45](#)
 controlling tell messages [12-1/12-5](#)
 controlling TERM objects [10-1/10-43](#)
 functional categories of [1-2](#), [7-1](#)
 object states, relation to [7-1](#)
 objects, relation to [7-4](#)
 PATHCOM format [7-5](#)
 Comments in command line [7-5](#)
 Communication
 between PATHMON environments
 PATHMON and TCP functions
 for [2-14](#)
 security requirements for [2-15](#)
 between Pathway environments
 commands to enable [6-9](#)
 demonstration of [6-7](#)

Communication protocol is not supported (message 3146) [13-32](#)

Compaq Tandem Performance Data Collector (TPDC)

See Tandem Performance Data Collector (TPDC)

Configuration

changing existing [4-13](#)

displaying information about [4-3](#)

examples [6-1](#)

Configuration file, for PATHMON environment

See PATHMON configuration file

Configuration limits and defaults [C-1/C-7](#)

Configured terminal (TERM) objects

See TERM objects

Configuring

a simple Pathway environment [6-1](#)

applications in a PATHMON environment [2-1](#)

limits for PATHMON environment [4-14](#)

multiple PATHMON environments [2-3](#)

objects for PATHMON environments

PROGRAM objects [2-12](#)

TCP objects [2-7](#)

TERM objects [2-11](#)

Conflict of data types during... (message 3184) [13-40](#)

Context data area, allocating [9-26](#)

CONTROL command

PATHMON [2-9](#)

TCP STATS [4-10](#)

Control file

See PATHMON configuration file

CONTROL TCP command [9-4/9-9](#)

CONTROL TERM command [10-7/10-8](#)

CONTROL-26 operation did not... (message 3175) [13-37](#)

CONTROL-26 protocol, Guardian [10-23](#), [11-19](#)

Conversational mode terminals [6-1](#)

COOL start option used in command files [6-6](#)

COUNT parameter

STATS TCP command [9-36](#)

STATS TERM command [10-33](#)

CPU\$ attribute

for TCP objects [2-7](#)

SET TCP command [9-21](#)

CPUs, switching primary and backup [4-16](#)

C-series to D-series software migrating [D-1/D-3](#)

D

Data area [5-5](#), [5-12](#)

Data left over on scatter... (message 3179) [13-39](#)

DBCS translation support not... (message 3061) [13-18](#)

DEBUG attribute SET TCP command [9-21](#)

Debugging [9-24](#), [10-13/10-14](#)

Debugging TERM objects [10-42](#)

DEFINE definitions [7-7](#)

Defining and adding objects

TCP objects [2-7](#)

TERM objects [2-11](#)

DELETE command [4-15](#), [4-25](#)

DELETE PROGRAM command [11-5](#)

DELETE TCP command [9-9](#)

DELETE TELL command [12-2](#)

DELETE TERM command [10-9](#)

DELETEDELAY attribute relation to dynamic server processes [5-4](#)

Deleting

PROGRAM objects [4-15](#)

SERVER objects [4-15](#)

TCPs [4-15](#)

TELL message text [4-25](#)

TERM objects [4-15](#)

Delimiter is not byte aligned (message 3182) [13-40](#)

DELINK queuedefinition of [5-2](#)statistics for [5-16](#), [5-17](#)**DEPENDING value is...**(message 3183) [13-40](#)**DEPENDING variable value too big**(message 3002) [13-3](#)**DETAIL parameter**STATS TCP command [9-36](#)STATUS TCP command [9-38](#)STATUS TERM command [10-35](#)**Device does not support...**(message 3065) [13-20](#)**Device does not support...**(message 3066) [13-20](#)**Device doesn't support...**(message 3060) [13-18](#)**Device requires intervention**(message 3042) [13-14](#)**Device subclass invalid**(message 3168) [13-36](#)DEVICEINFO procedure [10-26](#), [11-12](#)Devices [1-12](#), [1-17](#)Diagnosing problems, data to collect for [4-21](#), [4-24](#)DIAGNOSTIC attribute [10-20](#)DIAGNOSTIC-ALLOWED special register [10-20](#)Disk file names [7-7/7-8](#)DISPLAY BASE operation [F-1](#)DISPLAY statement [10-19](#)**Displaying**configuration information [4-3](#)statistics information [4-10](#)status information [4-6](#)**DISPLAY-PAGES attribute**how to calculate value for [F-1/F-4](#)SET PROGRAM command [11-17](#)SET TERM command [10-20](#)**DUMP attribute**CONTROL TCP command [9-5](#)SET TCP command [9-21](#)**Dumps**for PATHMON process [2-9](#), [4-16](#)for TCP [4-16](#), [4-22](#)forcing for TCP objects [4-22](#)D-series operating system [D-1/D-2](#)**E****ECHO attribute**SET PROGRAM command [11-18](#)SET TERM command [10-21](#)Echo mode [10-21](#)**Edit error occurred...**(message 3165) [13-35](#)**Edit error occurred...**(message 3175) [13-38](#)END-TRANS in WAIT display field [9-40](#)END-TRANSACTION statement [E-4](#)ERMBUF in WAIT display field [9-40](#)Error and event logging [4-16](#)

See also Log messages, Log files, and EMS

ERROR display field [9-39](#), [10-36](#)**Error dumps**for PATHMON process [2-9](#), [4-16](#)for TCP [4-16](#), [4-22](#)**Error during dialog verb execution**(message 3253) [13-59](#)Error during I/O to router (message 3140) [13-30](#)Error during I/O... (message 3231) [13-51](#)**Error during nowaited socket receive**(message 3150) [13-33](#)**Error during nowaited socket send**(message 3148) [13-33](#)**Error during server I/O**(message 3116) [13-28](#)**Error during server open**(message 3100) [13-23](#)**Error during terminal IO...**(message 3018) [13-9](#)**Error during terminal open**(message 3017) [13-8](#)

Error in backup... (message 3215) [13-46](#)
Error in dialog mode (message 3250) [13-58](#)
Error in non-retryable linkmon connect (message 3247) [13-56](#)
Error in retryable linkmon error (message 3248) [13-57](#)
Error in server class (message 3246) [13-56](#)
Error in TCP (message 3249) [13-57](#)
Error in XML generation (message 3149) [13-33](#)
Error when dialog is invalid due to primary TCP failure (message 3252) [13-59](#)
Error when dialog is not ended by server (message 3251) [13-59](#)
Error while creating session (message 3145) [13-32](#)
Errors
 application error action [11-20](#)
 diagnostic screens [10-20](#)
 screen recovery [9-27](#)
ERROR-ABORT attribute [11-20](#)
Event Management Service
 See EMS
Event Management Service (EMS)
 logging status and error information [4-16](#)
 viewing event messages for TCPs, TERM objects [4-18](#)
Exception conditions [4-17](#)
EXCLUSIVE attribute
 SET PROGRAM command [11-18](#)
 SET TERM command [10-22](#)
Expansion, file names [7-8](#)
Extended data segment
 See Data area, Code area
Extended memory for TCP [5-4](#)
External objects and processes
 LINKMON process [1-11](#)
 PATHCOM [1-11](#)
 TCPs [1-15](#)

External PATHMON name invalid (message 3227) [13-50](#)
External PATHMON processes [9-25](#)
External system name invalid (message 3228) [13-50](#)
External system name not defined (message 3229) [13-50](#)
External TCPs
 status, displaying [9-38](#)
External TCPs, viewing status of [4-7](#)

F

Field contains other than... (message 3067) [13-21](#)
Field contains other than... (message 3068) [13-21](#)
Field length exceeds... (message 3171) [13-36](#)
Field occurrence exceeds... (message 3185) [13-41](#)
Field-attribute entries area [F-1](#)
FILE attribute
 CONTROL TCP command [9-5](#)
 for TERM objects [2-11](#)
 INSPECT TERM command [10-13](#)
 SET TERM command [10-18](#)
File names Guardian [7-7/7-8](#)
File open error...file in use [9-34](#)
FILE parameter, RUN PROGRAM command [11-11](#)
File-identifier names [7-7](#)
File-system error during... (message 3214) [13-46](#)
File-system errors [13-2](#)
FlowMap product [4-21](#)
FREQTABLE option of STATS TERM command [5-25](#)
FREQTABLE parameter
 STATS TCP command [9-36](#)
 STATS TERM command [10-33](#)
Frequency distribution table displaying for TERM objects [5-25](#)
FROZEN state in WAIT display field [9-40](#)

Function unimplemented
(message 3203) [13-43](#)

G

Gateway program [1-12](#)
GDSX product [1-17](#)
Graphical user interfaces (GUIs) [1-15](#)
GUARDIAN-LIB attribute SET TCP
command [9-23](#)
GUARDIAN-LIB incompatible...
(message 3239) [9-31](#), [13-53](#)
GUARDIAN-SWAP attribute SET TCP
command [9-23](#)

H

HIGHPIN attribute SET TCP
command [9-24](#)
HIGHPIN attribute, for TCP objects [2-7](#)
HOMETERM attribute SET TCP
command [9-24](#)

I

IDS facility, for intelligent device
support [1-17](#)
IDS-related verbs [9-30](#)
IFPRICPU attribute PRIMARY TCP
command [9-13](#)
Illegal ACCEPT variable...
(message 3037) [13-13](#)
Illegal CPU number (message 1095) [9-8](#)
Illegal data reference (message 3008) [13-5](#)
Illegal delay value (message 3074) [13-23](#)
Illegal TCLPROG code file
(message 3104) [13-25](#)
Illegal TCLPROG directory file
(message 3103) [13-24](#)
Illegal terminal IO protocol word
(message 3014) [13-7](#)
Illegal terminal type specified...
(message 3011) [13-6](#)
Illegal timeout value (message 3169) [13-36](#)
IMON-I/O in WAIT display field [9-40](#)

INFO command
 attribute display with [4-3](#)
INFO display field [9-39](#), [10-36](#)
INFO PROGRAM command [4-6](#), [11-6/11-7](#)
INFO TCP command [4-4](#), [9-10/9-12](#)
INFO TELL command [4-25](#), [12-3](#)
INFO TERM command [4-5](#), [10-10/10-12](#)
INITIAL attribute
 for TERM objects [2-11](#)
 SET PROGRAM command [11-16](#)
 SET TERM command [10-19](#)
 START TERM command [10-30](#)
Initialization files [D-2](#)
Input-output devices and processes [2-10](#)
INSPECT attribute
 SET TCP command [9-24](#)
 SET TERM command [10-22](#)
Inspect breakpoint table full
(message 3222) [13-49](#)
Inspect not enabled for TCP
(message 3220) [13-48](#)
INSPECT TERM command [10-13/10-14](#)
Inspect terminal table full
(message 3221) [13-48](#)
INSPECT- in WAIT display field [9-40](#)
INST-*nnn* display fields [10-37](#)
Insufficient TERMPPOOL for request
(message 3073) [13-23](#)
Intelligent devices [1-17](#), [10-25](#)
Intelligent terminals, messages for [12-4](#)
Interactive mode [7-6](#)
Internal error in terminal format...
(message 3010) [13-5](#)
INTERVAL parameter
 STATS TCP command [9-36](#)
 STATS TERM command [10-33](#)
Invalid end of message character...
(message 3170) [13-36](#)
Invalid format message received...
(message 3200) [13-42](#)
Invalid I/O protocol value...
(message 3055) [13-16](#)

Invalid Katakana... (message 3062) [13-19](#)
 Invalid Katakana... (message 3063) [13-19](#)
 Invalid numeric item...
 (message 3040) [13-13](#)
 Invalid numeric item...
 (message 3053) [13-15](#)
 Invalid printer specification
 (message 3041) [13-14](#)
 Invalid pseudocode detected...
 (message 3001) [13-2](#)
 Invalid subscript value
 (message 3003) [13-3](#)
 IO protocol denied...
 (message 3054) [13-16](#)
 IOPROTOCOL attribute
 SET PROGRAM command [11-19](#)
 SET TERM command [10-23](#)
 IS-ATTACHED attribute
 RUN PROGRAM command [11-11](#)
 SET PROGRAM command [10-24](#),
 [11-22](#)
 ITMACQ in WAIT display field [9-40](#)
 ITMCOMP in WAIT display field [9-40](#)
 I/O error on control-26...
 (message 3173) [13-37](#)
 I/O error on... (message 3059) [13-17](#)
 I/O error with IMON or Inspect
 (message 3224) [13-49](#)
 I/O error (message 3161) [13-33](#)
 I/O operations, requested by TERM
 objects [5-18/5-22](#)
 I/O requests and replies [9-28](#)

L

LIKE attribute
 SET PROGRAM command [11-20](#)
 SET TCP command [9-25](#)
 SET TERM command [10-23](#)
 Limits
 changing existing [4-14](#)
 configuration [C-2/C-7](#)

Link management [4-19](#)
 as TCP task [5-2](#)
 queues maintained for [5-2](#)
 suggestions for improved performance
 improve PATHMON
 performance [4-20](#)
 increase number of links [5-17](#)
 LINK queue
 definition of [5-2](#)
 statistics for [5-16](#), [5-17](#)
 Linker [4-19](#)
 LINKMON processes
 defined [1-11](#)
 link management tasks [1-18](#)
 statistics gathering, for servers [1-18](#)
 Links
 definition of [4-19](#)
 dissolution of [4-19](#), [5-4](#)
 establishment of [5-2](#)
 interpreting statistics about [5-16](#), [5-17](#)
 managed and owned by
 PATHMON [4-19](#)
 relation to SEND operations [4-19](#)
 requesting new [5-3](#)
 TCP resources required for [5-3](#)
 to server classes [9-26](#)
 to server processes [4-19](#), [9-26](#)
 Load balancing across TCPs [2-9](#)
 Local objects
 PATHMON process [1-10](#)
 SERVER objects [1-11](#)
 TCPs [1-12](#)
 TERM and PROGRAM objects [1-12](#)
 Log files [4-16](#)
 and TMF considerations [E-5](#)
 example of [6-5](#)
 viewing information in [6-5](#)

M

MAXEXTERNALTCPS attribute [8-2](#)

Maximum number of waited run...
(message 1148) [11-13](#)

Maximum reply size exceeds...
(message 3114) [13-27](#)

MAXINPUTMSGLEN attribute [9-25](#)

MAXINPUTMSGGS attribute

SET PROGRAM command [11-19](#)

SET TCP command [9-25](#)

SET TERM command [10-23](#)

MAXLINKS attribute

for SERVER objects [6-4](#)

MAXPATHCOMS attribute, for PATHMON
environments [6-3](#)

MAXPATHWAYS attribute, for TCP
objects [6-9](#), [9-25](#)

MAXPROGRAMS attribute, for PATHMON
environments [2-4](#), [8-2](#)

MAXREPLY attribute, for TCP objects [2-7](#),
[5-10](#), [9-25](#)

MAXSERVERCLASSES attribute

for TCP objects [2-7](#), [6-9](#)

SET TCP command [9-26](#)

MAXSERVERPROCESSES attribute

for TCP objects [2-7](#), [6-9](#)

SET TCP command [9-26](#)

MAXSERVERS attribute for SERVER
objects [5-10](#)

MAXTCPS attribute, for PATHMON
environments [2-4](#), [8-2](#)

MAXTELLQUEUE attribute [8-3](#), [12-5](#)

MAXTELLS attribute [8-3](#), [12-5](#)

MAXTERMDATA attribute [2-7](#), [9-26](#)

MAXTERMDATA in TCP configuration...
(message 3232) [13-51](#)

MAXTERMS attribute

for PATHMON environments [2-4](#), [6-3](#)

for TCP objects [2-7](#), [5-9](#)

SET PATHWAY command [8-2](#)

SET TCP command [9-19](#)

MAXTMFRESTARTS attribute [8-3](#)

MAX-TPS, for peak transaction rate [5-11](#)

Measure product [4-21](#)

MEMMAN in WAIT display field [9-40](#)

Memory

allocation for terminals [10-21](#), [11-17](#)

swap [9-23](#)

Memory allocation error in router (message
3144) [13-32](#)

Memory dumps

for PATHMON process [2-9](#), [4-16](#)

for TCP [4-16](#)

Memory management

See also Code area, Data area

as TCP task [5-4](#)

for context data [5-12](#)

for SCREEN COBOL object code [5-12](#)

for terminal and server I/O buffers [5-4](#)

queues maintained for [5-5](#)

Message length exceeds...

(message 3172) [13-37](#)

Message to send exceeds...

(message 3167) [13-35](#)

Messages

general information [13-1/13-2](#)

Messages, error

additional information in [13-1/13-2](#)

operating system error numbers [13-2](#)

Messages, reply [9-25](#)

Messages, SCREEN COBOL [13-2](#)

Messages, TCP [13-2/13-55](#)

Messages, unsolicited

length, setting maximum [9-25](#)

protocol, specifying [10-23](#), [11-19](#)

queue, setting maximum [9-25](#), [10-23](#),
[11-19](#)

UMP I/O bytes allocated by TCP [9-28](#)

Migration, issues for [D-1/D-3](#)

Multiple unsolicited messages rejected
(message 3242) [13-55](#)

Multiple unsolicited messages rejected...
(message 3125) [13-30](#)

Multithreading

as feature of TCP [1-12](#)

definition of [1-12](#)

N

No room for another...

(message 3208) [13-44](#)

No room for new server class...

(message 3110) [13-26](#)

No room for new... (message 3230) [13-51](#)

No server process linked to

(message 3123) [13-30](#)

No space for new server process

(message 3119) [13-29](#)

No unsolicited message...

(message 3177) [13-38](#)

Node names [7-7](#)

Nondedicated devices and processes, and

PROGRAM objects [2-12](#)

Noninteractive mode [7-6](#)

NONSTOP attribute [9-27](#)

NonStop Kernel Open System Services
processes

See OSS processes

NonStop Kernel Open System Services
(OSS) [1-5](#)

NonStop Remote Server Call/MP (RSC/MP)
product

See NonStop RSC/MP product

NonStop RSC/MP product

clients, communicating with Pathway

servers [1-11](#)

for client/server computing [1-15](#)

in Pathway environment [1-6](#)

Not enough data for scatter...

(message 3180) [13-39](#)

NOWAIT option, and PROGRAM
objects [3-6](#)

NOWAIT parameter, RUN PROGRAM
command [11-13](#)

O

OBEYFORM parameter

INFO PROGRAM command [11-6](#)

INFO TCP command [9-11](#)

INFO TERM command [10-11](#)

Object libraries, for screen programs [2-10](#)

Objects

See also individual objects

altering definitions of [4-15](#)

and attributes [2-6](#)

controlling and maintaining,
overview [4-1](#)

defined

PATHMON process [1-10](#)

SERVER [1-11](#)

TCP [1-12](#)

defining and adding [2-6/2-14](#)

deleting [4-15](#)

descriptions of [7-1](#)

displaying information about [4-3/4-10](#)

external to PATHMON

environment [1-13](#)

PATHCOM commands for [7-4](#)

starting [3-1/3-7](#)

states [4-9](#), [7-1](#)

stopping [3-7/3-9](#)

Open of file to backup failed

(message 3213) [13-46](#)

Open System Services (OSS)

operating environment [1-5](#)

server classes defined [1-5](#), [1-11](#)

Operating system error numbers [13-2](#)

Optional attributes

PROGRAM objects [2-13](#)

TCPs [2-10](#)

TERM objects [2-12](#)

OUT attribute

- INFO PROGRAM command [11-6](#)
- INFO TCP command [9-10](#)
- INFO TELL command [12-3](#)
- INFO TERM command [10-10](#)
- SHOW TCP command [9-32](#)
- SHOW TERM command [10-28](#)
- STATS TCP command [9-35](#)
- STATS TERM command [10-32](#)
- STATUS TCP command [9-38](#)
- STATUS TERM command [10-35](#)

OUT parameter, SHOW PROGRAM command [11-24](#)

Overlay screen displayed in...
(message 3013) [13-7](#)

OWNER attribute

- for PROGRAM objects [2-14](#)
- SET PROGRAM command [11-21](#)

P

Parameter reference exceeds...
(message 3238) [13-53](#)

Parameters

- configuration limits for [C-1/C-7](#)
- for TCP... (message 3106) [13-25](#)

PATHCOM

- command and object summary [7-4](#)
- command format [7-5](#)
- command list [7-2/7-4](#)
- defined [1-5](#)
- modes [7-6/7-7](#)
- multiple processes for improved performance [3-5](#), [3-9](#)
- multiple processes for multiple users [1-11](#)
- overview [7-1](#)
- requester limit [9-34](#)
- reserved word list [B-1/B-3](#)
- returning the prompt [7-6](#)

PATHCOM commands

- ABORT TERM [3-9](#)
- ADD PROGRAM [2-12](#)
- ADD TCP [2-7](#)
- ADD TERM [2-11](#)
- ALTER [4-15](#)
- CONTROL TCP STATS [4-10](#)
- INFO PROGRAM [4-6](#)
- INFO TCP [4-4](#)
- INFO TERM [4-5](#)
- PRIMARY [4-16](#)
- RUN PROGRAM [3-5](#)
- SET PROGRAM [2-12](#)
- SET TCP [2-7](#)
- SET TCP STATS [4-10](#)
- SET TERM [2-11](#)
- SHUTDOWN [3-10](#)
- SHUTDOWN2 [3-10](#)
- START TCP [3-4](#)
- START TERM [3-4](#)
- STATS TCP [4-11](#)
- STATS TERM [4-13](#)
- STATUS TCP [4-7](#)
- STATUS TERM [4-8](#)
- STOP TCP [3-8](#)
- STOP TERM [3-8](#)
- SWITCH [4-16](#)
- TELL [4-24](#)

PATHCTL configuration file [7-7](#)

PATHMON configuration file

- description of [7-7](#)
- displaying contents of [4-3](#)
- effect of online changes on [4-14](#)
- recording changes to TCPs [9-4/9-9](#)

PATHMON environment

- See also Communication between PATHMON environments, PATHMON configuration file, Configuring applications [2-1](#)
- attributes for [2-4](#)

PATHMON environment (continued)

- configuration file [7-7](#)
- configuring across CPUs [1-8](#)
- controlling and maintaining [4-1/4-25](#)
- distributed applications [2-3/2-4](#)
- distributing applications [2-1](#)
- restarting [6-5](#)
- shutting down [3-10](#)

PATHMON object

- subtype 30 process support [11-13](#)

PATHMON process

- attributes for DUMP [2-9](#)
- communicating with an external TCP [1-15/1-16](#)
- defined [1-5](#)
- displaying information about [4-1/4-3](#)
- environment, configuring and managing objects in [1-19](#)
- error and event logging [4-16](#)
- error dumps [2-9](#)
- functions of [1-10](#)
- object definition of [1-10](#)
- object external [9-25](#)
- suggestions for improving performance [4-20](#)
- switching primary and backup CPUs for [4-16](#)
- TERM restart attempts [10-19](#)

PATHMON- in WAIT display field [9-40](#)**Pathsend processes**

- as requesters [1-11](#), [1-18](#)

PATHTCP2 cannot execute... (message 3243) [13-55](#)**PATHTCP2 code file [9-1](#)****PATHTCPL [9-23](#)**

- See also TCP user library object file

Pathway environment

- control and operation commands [8-1/8-4](#)
- described [1-1](#)

- PATHMON object, definition of [1-10](#)

PATHWAY object, SET command [8-1/8-4](#)**Pathway subsystem, configuration limits [C-1/C-7](#)****Pathway translation server [1-11](#)****Peak transaction rate, calculating [5-11](#)****PENDING display field [9-40](#)****Personal computers [1-15](#)****PINs [9-24](#), [D-1](#)****PMCB-LOCK in WAIT display field [9-40](#)****POBJ file, collecting for TCP problem diagnosis [4-23](#)****Pool space, TCP [9-30](#)****POWERONRECOVERY attribute [9-27](#)****PRI attribute, SET TCP command [9-27](#)****PRI run option [2-8](#)****PRIMARY command [4-16](#)****PRIMARY TCP command [9-13/9-14](#)****PRINTER attribute**

- RUN PROGRAM command [11-11](#)

- SET PROGRAM command [11-21/11-22](#)

- SET TERM command [10-23](#)

Printer I/O error (message 3043) [13-14](#)**Problem diagnosis, data to collect for [4-21](#), [4-24](#)****PROCESS attribute, for TCP objects [2-7](#)****PROCESS attribute, SET TCP command [9-27](#)****Process creation failure... (message 1038) [9-31](#)****Process identifier (PIN) [D-1](#)****Process priority, setting [2-8](#)****Processor does not have... (message 3209) [13-45](#)****Production environment, ensuring availability of [4-24](#)****PROGRAM [2-12](#)****PROGRAM attribute**

- SET TCP command [9-28](#)

PROGRAM objects

- ADD command [11-2](#)

- ALTER command [4-15](#), [11-3/11-4](#)

PROGRAM objects (continued)

- as templates for device types [2-14](#)
 - configuring [2-12/2-14](#)
 - defined [1-5](#), [1-12](#), [2-10](#)
 - defining and adding [2-12/2-14](#)
 - DELETE command [4-15](#), [11-5](#)
 - displaying information about [4-6](#)
 - ERROR-ABORT [2-13](#)
 - INFO command [11-6/11-7](#)
 - OWNER [2-14](#)
 - RESET command [11-8/11-9](#)
 - RUN command [11-10/11-14](#)
 - running [3-5](#)
 - SECURITY [2-14](#)
 - security for [2-14](#)
 - SET command [11-15/11-23](#)
 - SHOW command [11-24/11-25](#)
 - stopping [3-7](#)
 - terminal types [11-12](#)
 - unsolicited messages, specifying [11-19](#)
- PROGRAM unit name invalid
(message 3044) [13-14](#)
- Program unit or SCOBOL...
(message 3058) [13-17](#)
- PROGRAM-ID sentence [10-19](#)
- Prompts, command [7-6](#)
- Pseudocode size too big
(message 3023) [13-10](#)
- PU-nnn display fields [10-37](#)

Q**Queues**

- for checkpointing [5-6](#), [5-14](#)
- for link management [5-2](#), [5-14](#)
- for memory management [5-5](#), [5-14](#)
- for TERMPOOL and
SERVERPOOL [5-14](#)

R

- READY in WAIT display field [9-40](#)
- Received message exceeds...
(message 3166) [13-35](#)
- Received message larger than...
(message 3163) [13-34](#)
- Received message smaller than...
(message 3162) [13-34](#)
- Reconfiguring Pathway applications
 - a scenario [4-2](#)
 - adding, altering, and deleting
objects [4-14](#)
 - effect of changes [4-3](#)
 - specifying new limits [4-14](#)
- RECOVERING display field [9-40](#)
- Recovery of screens [9-27](#)
- Referenced screen is illegal for...
(message 3009) [13-5](#)
- REFRESH-CODE TCP
command [9-14/9-16](#)
- Reply number not known...
(message 3112) [13-27](#)
- REQ CNT, for TCP buffer allocation [5-11](#)
- Request invalid for server state
(message 3118) [13-28](#)
- Request not allowed while...
(message 3223) [13-49](#)
- Request pending (message 3207) [13-44](#)
- Requested device color/highlight...
(message 3071) [13-22](#)
- Requested function not supported...
(message 3226) [13-50](#)
- Requester limit error [9-34](#)
- Requesters, Pathsend and SCREEN
COBOL [1-18](#)
- Requesting a new link [5-3](#)
- Required attributes
 - PROGRAM objects [2-13](#)
 - TERM objects [2-11](#)
- Reserved word list [B-1/B-3](#)

RESET parameter

ALTER PROGRAM command [11-3](#)ALTER TCP command [9-3](#)ALTER TERM command [10-5](#)STATS TCP command [9-36](#)STATS TERM command [10-33](#)RESET PROGRAM command [11-8/11-9](#)RESET TCP command [9-16/9-17](#)RESET TERM command [10-15](#)

Response time

displaying information about [5-24](#), [5-25](#)possibilities for decreased [4-3](#)reasons for increased [4-3](#)

suggestions for improved

determine efficient size for data transfers [5-21](#)reduce number of I/O operations required [5-21](#), [5-22](#)remove the TCLPROG file [4-21](#)set CHECK-DIRECTORY attribute [4-21](#)specify sufficient buffer space [4-3](#), [5-9](#), [5-22](#)specify sufficient code area [5-13](#)start SERVER objects before TCPs [3-3](#)use external TCPs [4-20](#)Restarting Pathway environment, example of [6-5](#)RESTART-COUNTER special register [10-17](#)RESUME TERM command [10-16/10-17](#)Router cannot handle more TCPs... (message 3142) [13-31](#)Router cannot handle more TERMS... (message 3143) [13-31](#)

Router process

defined [1-5](#)

Router processes

defined [1-13](#)relationship to Pathway environment [1-5](#)

Router processes (continued)

starting [3-1](#)stopping [3-7](#)RUN command to start PROGRAM objects [3-5](#)Run priority, specifying [9-27](#)RUN PROGRAM command [3-5](#), [11-10/11-14](#)Run-time attribute setting invalid... (message 3072) [13-23](#)**S**

Screen cache error...

(message 3056) [13-16](#)

SCREEN COBOL

block-mode, running programs for [11-23](#)collecting data about for problem diagnosis [4-23](#)collecting source code for TCP problem diagnosis [4-23](#)directory file, checking [9-19](#)initial program unit, setting [10-19](#), [10-30](#), [11-16](#)I/O bytes, allocated by TCP [9-28](#)messages [13-2](#)object libraries [2-7](#)object library file [9-19](#), [10-24](#), [11-19](#)ON ERROR clause [8-3](#)program debugging [10-13/10-14](#)program units [2-7](#)TCP, naming the [11-15](#)temporary terminals, startup on [11-10/11-14](#)terminal types [11-12](#)SCREEN COBOL compiler [6-2](#)Screen diagnostics [10-20](#)Screen operation done without... (message 3007) [13-4](#)

Screen programs

- coding [6-2](#)
- configuring TERM and PROGRAM objects for [2-10](#)
- executed by TCPs [1-12](#)
- names of [2-11](#)
- object libraries for [2-7](#), [2-10](#)
- source code for examples [G-1](#)
- user conversion procedures for [2-10](#)

Screen recovery executed...

(message 3004) [13-3](#)

Screen recovery, setting [9-27](#)

Screen referenced but not...

(message 3012) [13-7](#)

Screen-image area [F-1](#)

Security

- for PROGRAM objects [2-13](#)
- requirements for communication between PATHMON environments [2-15](#)
- specifying for PROGRAM objects [2-14](#)

SECURITY attribute

- for PROGRAM objects [2-13](#), [2-14](#)
- SET PROGRAM command [11-22](#)

SEND message verb, specifying

maximum [9-25](#)

SEND operations and links [4-19](#)SEND statement [E-4](#)

SEND, error during I/O...

(message 3231) [13-51](#)

SEND, external PATHMON name invalid

(message 3227) [13-50](#)

SEND, external system name invalid

(message 3228) [13-50](#)

SEND, external system name not defined

(message 3229) [13-50](#)

SEND, no room for new...

(message 3230) [13-51](#)

Send, server class name invalid

(message 3022) [13-10](#)

SERVBUF in WAIT display field [9-40](#)

Server class undefined

(message 3117) [13-28](#)

Server classes

See also SERVER objects

maximum TCP links [9-26](#)

SERVER display field [10-37](#)SERVER in WAIT display field [9-40](#)

SERVER objects

altering [4-15](#)

and SCREEN COBOL requesters [1-18](#)

defined [1-5](#), [1-11](#)

definition of [1-11](#)

deleting [4-15](#)

Server process unknown

(message 3233) [13-52](#)

Server processes

links to [2-14](#), [4-19](#)

maximum TCP links [9-26](#)

Server programs, source code for [G-5](#)Server reply messages [9-25](#)SERVERIO in WAIT display field [9-40](#)SERVERPOOL attribute [9-28](#)

SERVERPOOL, TCP storage pool

configuration attribute [2-7](#)

description of [5-4](#)

determining effective size for [5-10/5-11](#)

statistics for [5-8](#), [5-9](#)

Service provider, collecting problem data

for [4-21](#), [4-24](#)

Session is null (message 3147) [13-32](#)SET PATHWAY command [8-1/8-4](#)SET PROGRAM command [2-12](#),
[11-15/11-23](#), [E-1](#)SET TCP command [2-7](#), [9-18/9-31](#)SET TCP STATS command [4-10](#)SET TERM command [2-11](#), [10-18/10-27](#),
[E-1](#)Shared access [10-22](#), [11-18](#)SHOW PROGRAM command [11-24/11-25](#)SHOW TCP command [9-32/9-33](#)SHOW TERM command [10-28/10-29](#)SHUTDOWN command [3-10](#)SHUTDOWN2 command [3-10](#)

Shutting down Pathway

See also SHUTDOWN2 command

Slots [9-26](#)

SNAX High-Level Support
(SNAX/HLS) [1-17](#)

Socket error (message 3141) [13-31](#)

Source code

screen program example [G-1](#)

server program example [G-5](#)

Special register [3-11](#)

specifies [9-38](#)

SPI, using to monitor event messages [4-18](#)

START command, guidelines [3-2](#)

START TCP command [3-4](#), [9-33/9-34](#)

START TERM command [3-4](#), [10-30/10-31](#)

Starting objects

PROGRAM objects with RUN
command [3-5](#)

TCPs [3-4](#)

TERM objects [3-4](#)

Starting router processes [3-1](#)

Startup, cool

CONTROL TCP command, effect
of [4-22](#)

Startup, cool, CONTROL TCP command,
effect of [9-8](#)

STATE display field [9-39](#)

STATE option, with STATUS TCP
command [4-7](#)

STATE parameter

ABORT TERM command [10-2](#)

INFO TCP command [9-10](#)

INFO TERM command [10-10](#)

REFRESH-CODE TCP command [9-15](#)

RESUME TERM command [10-16](#)

START TCP command [9-33](#)

START TERM command [10-30](#)

STATS TCP command [9-36](#)

STATS TERM command [10-33](#)

STATUS TCP command [9-38](#)

STATUS TERM command [10-35](#)

STATE parameter (continued)

STOP TCP command [9-43](#)

STOP TERM command [10-40](#)

SUSPEND TERM command [10-42](#)

SWITCH TCP command [9-44](#)

TELL TERM command [12-4](#)

Statistics

See also TCP statistics, TERM
statistics, STATS command

collected by

TCPs [5-6](#)

collected by TCPs [4-10](#)

displaying information about [4-10](#)

for links [5-16](#), [5-17](#)

for TCPs [4-11](#), [5-7/5-17](#)

for TERM objects [4-13](#), [5-17/5-26](#)

STATS attribute

CONTROL TCP command [9-7](#)

SET TCP command [9-28](#), [9-29](#)

STATS TCP command [4-11](#), [9-35/9-37](#)

STATS TERM command [4-13](#), [10-32/10-34](#)

STATUS TCP command [4-7](#), [9-37/9-42](#)

STATUS TERM command [4-8](#),
[10-35/10-39](#)

Status, displaying information [4-6/4-10](#)

STOP TCP command [3-8](#), [9-42/9-43](#)

STOP TERM command [3-8](#), [10-40/10-41](#)

STOPMODE display field [9-40](#)

STOPMODE register [3-11](#)

Stopping

PATHMON-controlled objects [3-7](#)

Pathway environment

See SHUTDOWN2 command

router processes [3-7](#)

Stopping execution of commands [7-6](#)

Storage pools

See SERVERPOOL, TCP storage pool,
TERMPool

Subtype 30 processes [11-13](#)

SUSPEND TERM command [10-42/10-43](#)

SUSPEND-ABORT state [10-19](#)

SUSPEND-RESUME state [10-19](#)
 SUSP- in WAIT display field [9-40](#)
 SWAP attribute [9-29](#)
 Swap file creation error
 (message 3107) [13-26](#)
 Swap file I/O error (message 3109) [13-26](#)
 Swap file open error
 (message 3108) [13-26](#)
 Swap files [5-14](#), [9-23](#), [9-29](#)
 SWITCH command [4-16](#)
 SWITCH TCP command [9-44/9-45](#)
 SYNCID violation... (message 3201) [13-42](#)

T

TACL commands, no embedded comments
 in scripts [7-5](#)
 TACL scripts [7-5](#)
 Takeover by backup (message 3218) [13-48](#)
 Tandem Performance Data Collector
 (TPDC) [4-21](#)
 Task already using another...
 (message 3225) [13-49](#)
 TCLPROG attribute
 for TCP objects [2-7](#)
 SET PROGRAM command [11-19](#)
 SET TCP command [9-19](#)
 SET TERM command [10-24](#)
 TCLPROG code file error
 (message 3122) [13-29](#)
 TCLPROG code file open...
 (message 3102) [13-24](#)
 TCLPROG directory entry is bad
 (message 3024) [13-10](#)
 TCLPROG directory file error
 (message 3121) [13-29](#)
 TCLPROG directory file open...
 (message 3101) [13-24](#)
 TCLPROG file [4-21](#)
 TCP attribute
 for PROGRAM objects [2-13](#)
 for TERM objects [2-12](#)
 SET PROGRAM command [11-15](#)

TCP attribute (continued)
 SET TERM command [10-19](#)
 TCP cannot handle...
 (message 3206) [13-44](#)
 TCP experiencing persistent...
 (message 3237) [13-52](#)
 TCP internal error (message 3216) [13-47](#)
 TCP memory dump not taken (message
 3197) [9-8](#)
 TCP memory dump not taken
 (message 3197) [13-41](#)
 TCP memory dump taken
 (message 3219) [13-48](#)
 TCP messages [13-2/13-55](#)
 TCP objects
 ADD command [9-2](#)
 ALTER command [9-3/9-4](#)
 backup processes [9-27](#)
 code area, allocating [9-21](#)
 CONTROL command [9-4/9-9](#)
 defined [1-5](#)
 DELETE command [9-9](#)
 INFO command [9-10/9-12](#)
 naming [9-27](#), [11-15](#)
 pool space [9-30](#)
 PRIMARY command [9-13/9-14](#)
 REFRESH-CODE command [9-14/9-16](#)
 RESET command [9-16/9-17](#)
 SET command [9-18/9-31](#)
 SHOW command [9-32/9-33](#)
 START command [9-33/9-34](#)
 STATS command [9-35/9-37](#)
 STATUS command [9-37/9-42](#)
 STOP command [9-42/9-43](#)
 SWITCH command [9-44/9-45](#)
 TMF restarts, specifying maximum [8-3](#)
 TCP state does not allow...
 (message 3204) [13-43](#)
 TCP statistics [5-7/5-17](#)
 AREA INFO [5-12](#)
 CONTROL TCP STATS command [4-10](#)

TCP statistics (continued)

POOL INFO [5-8](#)QUEUE INFO [5-14](#)SET TCP STATS command [4-10](#)TCP tasks [5-1/5-6](#)checkpointing [5-6](#)link management [5-2](#)memory management [5-4](#)statistics gathering [5-6](#)TCP trap (message 3217) [13-47](#)TCP user library object file [2-10](#)TCP2 code file [9-1](#)

TCPs

altering [4-15](#)as external objects [1-15](#)as multitasking processes [2-6](#)as multithreaded process [1-12](#)

attributes for

BACKUPCPU [4-16](#)CODEAREALEN [2-7](#)CPUS [2-7](#)DUMP [4-16](#)HIGHPIN [2-7](#)MAXPATHWAYS [6-9](#)MAXREPLY [2-7](#)MAXSERVERCLASSES [2-7](#), [6-9](#)MAXSERVERPROCESSES [2-7](#),
[6-9](#)MAXTERMDATA [2-7](#)MAXTERMS [2-7](#)optional [2-7](#)PROCESS [2-7](#)SERVERPOOL [2-7](#)TCLPROG [2-7](#)TERMBUF [2-7](#)TERMPPOOL [2-7](#)collecting data about for problem
diagnosis [4-21](#)configuring [2-7/2-10](#)defined [1-5](#), [1-12](#)

TCPs (continued)

defining and adding [2-7/2-10](#)deleting [4-15](#)detecting exception conditions [4-17](#)

displaying information about

configuration [4-4](#)statistics [4-11](#)status [4-7](#)distributing transaction load across [2-9](#)error dumps [4-22](#)sharing links [4-19](#)starting [3-4](#)before TERM objects [3-3](#)in parallel for improved
performance [3-5](#)stopping [3-8](#)stopping in parallel for improved
performance [3-9](#)switching primary and backup CPUs
for [4-16](#)tasks performed by [2-6](#), [5-1](#)TELL command [4-24](#)TELL messages [4-24](#), [4-25](#)

TELL objects

circular queue [12-5](#)DELETE command [12-2](#)INFO command [12-3](#)TELL TERM command [12-4/12-5](#)TELL TERM command [12-4/12-5](#)TELL-ALLOWED special register [12-1](#)

Templates, for input-output devices

See PROGRAM objects

Temporary devices and processes

and PROGRAM objects [3-5](#)starting objects for [3-5](#)

Temporary terminal (PROGRAM) objects

See PROGRAM objects

TERM cannot call PU with...
(message 3036) [13-13](#)

TERM objects

ABORT command [10-2/10-3](#)
 ADD command [10-4](#)
 ALTER command [10-5/10-6](#), [10-7/??](#)
 altering [4-15](#)
 attributes for
 BREAK [2-12](#)
 DIAGNOSTIC [2-12](#)
 FILE [2-11](#)
 INITIAL [2-11](#)
 optional [2-12](#)
 required [2-11](#)
 TCP [2-12](#)
 TYPE [2-12](#)
 Break key for [2-12](#)
 collecting data about for problem diagnosis [4-23](#)
 configuring [2-10/2-12](#)
 defined [1-5](#), [1-12](#), [2-10](#)
 defining and adding [2-11/2-12](#)
 DELETE command [4-15](#), [10-9](#)
 detecting exception conditions for [4-17](#)
 displaying information about
 configuration [4-5](#)
 statistics [4-13](#)
 status [4-8](#)
 distributing across TCPs [2-9](#)
 echo mode, overriding [10-21](#)
 implicit creation of [3-5](#)
 INFO command [10-10/10-12](#)
 INSPECT command [10-13/10-14](#)
 naming [10-18](#)
 RESET command [10-15](#)
 restart attempts, specifying [10-19](#)
 RESUME command [10-16/10-17](#)
 SET command [10-18/10-27](#)
 SHOW command [10-28/10-29](#)
 START command [10-30/10-31](#)
 starting [3-4](#)

TERM objects (continued)

starting in parallel for improved performance [3-5](#)
 STATS command [10-32/10-34](#)
 STATUS command [10-35/10-39](#)
 STOP command [10-40/10-41](#)
 stopping [3-8](#)
 stopping in parallel for improved performance [3-9](#)
 SUSPEND command [10-42/10-43](#)
 TELL command [12-4/12-5](#)
 unsolicited messages, specifying [10-23](#)
 TERM statistics [5-17/5-26](#)
 AREA INFO [5-23](#)
 frequency distribution [5-25](#)
 I/O INFO [5-18](#)
 RESPONSE TIME INFO [5-24](#)
 TERM type not defined... (message 1086) [11-13](#)
 TERMBUF attribute [9-29](#)
 TERMBUF attribute, for TCP objects [2-7](#), [5-9](#)
 Terminal identifier not known... (message 3202) [13-43](#)
 Terminal input data stream... (message 3025) [13-11](#)
 TERMINAL IS clause [10-26](#)
 Terminal stack space overflow (message 3016) [13-8](#)
 Terminal state does not... (message 3205) [13-44](#)
 Terminal stopped... (message 3050) [13-15](#)
 Terminal stopped... (message 3052) [13-15](#)
 Terminal suspended... (message 3051) [13-15](#)
 Terminals [6-1](#)
 See also TERM objects
 application errors [11-20](#)
 context data [9-26](#)
 debugging information, specifying [9-24](#)
 display pages, setting [10-20](#), [11-17](#)
 echo mode, overriding [10-21](#)

Terminals (continued)

- memory area [10-21](#), [11-17](#)
- output buffers, allocating [9-29/9-30](#)
- SCREEN COBOL programs [11-23](#)
- specifying maximum [9-19](#)
- specifying type [10-25](#)
- trailing blanks, suppressing [10-25](#), [11-20](#)
- type, specifying [11-16](#)
- user [2-6](#)

Terminals, see TERM objects and PROGRAM objects

TERMINAL-PRINTER special register [10-24](#), [11-11](#), [11-21](#)

Terminating PATHCOM commands [7-6](#)

TERMINATION-SUBSTATUS special register [10-19](#), [11-16](#)

TERMPOOL attribute [2-7](#), [9-30](#)

TERMPOOL, TCP storage pool

- description of [5-4](#)
- determining effective size for [5-9](#)
- statistics for [5-8](#), [5-9](#)

TERMREAD in WAIT display field [9-40](#)

TERMWRITE in WAIT display field [9-40](#)

TERM-OPEN in WAIT display field [9-40](#)

TERM^QUEUE^FULL error [10-23](#), [11-19](#)

Threads [1-12](#)

Timeout error messages (3161), filtering [9-28](#)

TIMEOUT in WAIT display field [9-40](#)

TMF attribute

- SET PROGRAM command [11-23](#)
- SET TERM command [10-25](#)

TMF not configured (message 3030) [13-12](#)

TMF not running (message 3031) [13-12](#)

TMF TFILE open failure... (message 3032) [13-12](#)

TMF (Transaction Management Facility)

- precautions for using parameters [E-5](#)
- restarts, specifying maximum [8-3](#)
- TERM activity, resuming [10-17](#)

To [3-8](#)

Too many PROGRAM entries (message 1108) [11-4](#)

Too many TERM entries (message 1101) [10-6](#)

TPDC

See Tandem Performance Data Collector (TPDC)

TRAILINGBLANKS attribute

- SET PROGRAM command [11-20](#)
- SET TERM command [10-25](#)

Transaction I/O error... (message 3028) [13-11](#)

Transaction Management Facility (TMF)
See TMF (Transaction Management Facility)

Transaction message send failure (message 3021) [13-9](#)

Transaction message size exceeds... (message 3113) [13-27](#)

Transaction mode violation (message 3027) [13-11](#)

Transaction reply size invalid (message 3115) [13-27](#)

Transaction restart limit reached (message 3029) [13-12](#)

Transaction workload

- calculating [5-11](#)
- distributing across TCPs [2-9](#)

TRANSACTION-ID special register [10-25](#), [11-23](#)

Translation server, Pathway [1-11](#)

TRANSMODE display field [9-40](#)

TRANSMODE register [3-11](#)

TRANS-nnn display fields [10-37](#)

Truncation occurred during... (message 3064) [13-19](#)

TYPE attribute

- for PROGRAM objects [2-13](#)
- for TERM objects [2-12](#)
- RUN PROGRAM command [11-12](#)
- SET PROGRAM command [11-16](#)
- SET TERM command [10-25](#)

U

Unilateral abort... (message 3069) [13-22](#)

Unsolicited message rejected...
(message 3241) [13-54](#)

User conversion procedures, for screen
programs [2-10](#)

User conversion routines, collecting for TCP
problem diagnosis [4-23](#)

User terminals [2-6](#)

V

Value for MAXINPUTMSG...
(message 3240) [13-53](#)

Variable field size would exceed...
(message 3181) [13-39](#)

Volume names [7-7](#)

W

WAIT display field [9-40](#)

WAIT parameter, STOP TCP
command [9-42](#)

Wild Card Support

 PROGRAM commands [11-25](#)

 TCP commands [9-45](#)

 TERM commands [10-43](#)

Workload, distributing across TCPs [2-9](#)

Wrong transfer count in...
(message 3019) [13-9](#)

Special Characters

! parameter

 SUSPEND TERM command [10-42](#)

\$SYSTEM.SYSTEM.PATHTCP2 [9-1](#), [9-28](#)

\$SYSTEM.SYSTEM.PATHTCPL [9-23](#)

* (Asterisk)

 using as a wild card in a
 command [3-3/3-4](#), [3-8](#)

Content Feedback

First Name: _____
Phone: _____
Company: _____

Last Name: _____
e-mail address: _____

(All contact information fields are required.)

If you're reporting an error or omission, is your issue:

- ☐ **Minor:** I can continue to work, but eventual resolution is requested.
- ☐ **Major:** I can continue to work, but prompt resolution is requested.
- ☐ **Critical:** I cannot continue to work without immediate response.

Comments (give sufficient detail to help us locate the text):

Thank you for taking the time to provide us with your comments.

You can submit this form online, e-mail it as an attachment to pubs.comments@hp.com, fax it to 408-285-5520, or mail it to:

Hewlett-Packard Company
NonStop Enterprise Division
19333 Vallco Parkway, MS 4421
Cupertino, CA 95014-2599
Attn.: Product Manager, Software Publications

