Hewlett Packard
Enterprise

TS/MP 2.7 System Management Manual

Part Number: 862347–003
Published: June 2018
Edition: L15.02 and all subsequent L-series RVUs and, J06.15 and all subsequent J-series RVUs

Notices

The information contained herein is subject to change without notice. The only warranties for Hewlett Packard Enterprise 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. Hewlett Packard Enterprise shall not be liable for technical or editorial errors or omissions contained herein.

Confidential computer software. Valid license from Hewlett Packard Enterprise 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.

Links to third-party websites take you outside the Hewlett Packard Enterprise website. Hewlett Packard Enterprise has no control over and is not responsible for information outside the Hewlett Packard Enterprise website.

Acknowledgments

Microsoft® and Windows® are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Intel®, Itanium®, Pentium®, Intel Inside®, and the Intel Inside logo are trademarks of Intel Corporation in the United States and other countries.

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.

UNIX® is a registered trademark of The Open Group.

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

Warranty

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 Enterprise 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 Informations system 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.

Contents

About This Manual

This manual provides guidelines and examples for configuring and managing the L15.02 and subsequent RVUs, J06.15 and subsequent RVUs, and H06.26 and subsequent RVUs of TS/MP, which run on HPE Integrity NonStop Servers and HPE Integrity NonStop BladeSystem. This manual is one of the set of manuals that describes the TS/MP and Pathway/iTS. The contents of these two products are as follows:

- NonStop TS/MP: This product consists of the PATHMON process, the application cluster services (ACS) subsystem processes, the PATHCOM process and interface, the PDMCOM process, the PDMI, and the Pathsend procedure calls.
- Pathway/iTS: This product consists of the terminal control process (TCP), the SCREEN COBOL compiler and run-time environment, and the SCREEN COBOL Utility Program (SCUP).

Together with the TMF and TMF subsystem, TS/MP forms the foundation for Hewlett Packard Enterprise's open transaction processing and client/server products.

The Pathway/iTS product supports requestor programs that run in the Guardian environment and communicate with terminals and intelligent devices. It requires the services of the NonStop TS/MP product.

Purpose of This Manual

This manual is both a task-oriented manual and a reference manual. It describes how to start, configure, and manage a PATHMON environment using the PATHCOM interactive management interface to NonStop TS/MP. It describes TACL commands and PATHCOM commands used to start, configure, and manage a PATHMON controlled environment on NonStop systems. This manual also includes information on monitoring and adjusting your PATHMON environment to optimize performance, diagnosing and fixing problems, as well as manageability guidelines such as how to start PATHMON-controlled objects in parallel to help performance.

It also provides syntax and complete descriptions of all PATHCOM commands and PATHCOM, PATHMON, and ACS subsystem messages.

If your environment includes Pathway/iTS, you must use this manual in conjunction with the **Pathway/iTS System Management Manual**, which provides information for configuring and controlling Pathway/iTS objects in a PATHMON environment. Read this manual first for an overview of the PATHMON environment and to find out how Pathway/iTS objects are supported.

Product Version

NonStop TS/MP 2.7

Supported Release Version Updates (RVUs)

This publication supports L15.02 and all subsequent L-series RVUs, J06.15 and all subsequent J-series RVUs, and H06.26 and all subsequent H-series RVUs, until otherwise indicated by the replacement publications.

Intended Audience

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

The task-oriented sections of this manual are also intended for individuals writing programs to manage server classes in a PATHMON environment programmatically. Such programmers also need the

reference information in the *TS/MP 2.5 Management Programming Manual*. Programmers using Pathway/iTS must also see the *Pathway/iTS Management Programming Manual*.

New and Changed Information

Changes to the 862347-003 manual

This version of the manual is applicable for J06.22 RVUs.

Changes to the 862347-002 manual

- Updated the cause and recovery information in error message 1516 in **PDMCOM Messages** section.
- Added the new error message 1538 in **PDMCOM Messages** section.

Changes to the 862347-001 manual

A new manual released for TS/MP 2.6 version.

Related Documentation

For more information specific to managing a PATHMON environment, see:

Manual	Description
Pathway/iTS System Management Manual	Provides instructions and guidelines for configuring and controlling the 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 runtime environment). This manual also provides syntax and complete descriptions of all PATHCOM commands, PATHMON error messages, and TCP error messages specific to the Pathway/iTS product.
TS/MP 2.5 Management Programming Manual	You need this manual if you are writing a management application to manage a PATHMON environment. It describes the management programming interface to the PATHMON environment as a whole and to the NonStop TS/MP objects in that environment. This interface includes the programmatic commands and responses, event messages, and related objects for NonStop TS/MP.
Pathway/iTS Management Programming Manual	You need this manual if you are writing a management application to manage the Pathway/iTS objects in a PATHMON environment. It describes the management programming interface to the Pathway/iTS objects in the PATHMON environment. This interface includes the programmatic commands and responses, event messages, and related objects for Pathway/iTS.

For information about informational, warning, and error messages, see:

Manual	Description
Operator Messages Manual	Describes system messages and provides an explanation of the cause, a discussion of the effect on the system, and suggestions for corrective action. The “NonStop TS/MP and Pathway/iTS Messages” section describes the operator messages generated by the PATHMON environment.
Guardian Procedure Errors and Messages Manual	Describes the Guardian messages for NonStop systems that use the NonStop operating system. The manual covers the error codes and error lists associated with Guardian procedure calls and the interprocess messages sent to application programs by the operating system and the command interpreter.

For information about programming server processes and Pathsend procedure calls, see the **TS/MP 2.5 Pathsend and Server Programming Manual**. For information about TS/MP 2.7, see *TS/MP 2.7 ACS Reference Manual*.

For information about other NonStop software products associated with Pathway transaction processing environments, see:

Manual	Description
Introduction to NonStop Transaction Processing	Describes the environment, components, and benefits of Hewlett Packard Enterprise transaction processing products, including TS/MP, Pathway/iTS, and related products such as TMF and the NonStop TUXEDO transaction processing system.
Guardian User's Guide	Provides basic information about the programs and utilities that are used most often by general system or application users. The guide addresses beginning users of NonStop systems.

Other Manuals in the Manual Set

Manuals	Description
TS/MP 2.5 Pathsend and Server Programming Manual	This manual describes the Pathsend procedure calls, included as part of TS/MP, and how to use those calls to write requestor programs. It also describes how to design and code Pathway servers for use with all types of requestors and clients.
TS/MP 2.7 ACS Reference Manual	This manual supplements the NonStop TS/MP manual set. It describes the new features provided by TS/MP 2.7; is intended for system managers, system operators, and application programmers who use Pathsend programming under the TS/MP 2.7 environment.

Additional Manuals

The following manuals might assist readers of this manual:

Manuals	Description
Availability Guide for Application Design	Describes the features of NonStop systems that support the availability of applications, including information about application availability in the Pathway environment.
Measure Reference Manual	Describes the Measure product for balancing and tuning NonStop systems. The manual covers commands, measurable entities, callable procedures, command error messages, and programmatic error codes.

Table Continued

Manuals	Description
Measure User's Guide	Describes how to use the Measure product for balancing and tuning NonStop systems. The manual illustrates how to configure measurements, collect performance data, instrument a user application, and so on.
HPE NonStop TMF Operations and Recovery Guide	Describes how to operate the TMF subsystem and recover from error conditions.
HPE NonStop TMF Reference Manual	Describes how to use the TMFCOM command interface to TMF.

Publishing History

Part Number	Product Version	Published
862347–003	NonStop TS/MP 2.7	June 2018
862347–002	NonStop TS/MP 2.7	August 2017
862347–001	NonStop TS/MP 2.6	June 2016

Introduction to NonStop TS/MP System Management

This section briefly describes the objects and processes in a Pathway online transaction processing (OLTP) environment. This section also introduces the tasks you must perform to manage a PATHMON environment, as well as applicable tools and commands.

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

- The NonStop TS/MP product
- Both the NonStop TS/MP and Pathway/iTS products

NOTE: This manual provides information specific to processes and commands associated with objects managed through the NonStop TS/MP product. Commands associated with TCP, TERM, and PROGRAM objects, which are provided by the Pathway/iTS product, are described in the *Pathway/iTS System Management Manual*. For information about TS/MP 2.6 and later, see the *TS/MP 2.7 ACS Reference Manual*.

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

The table presents a comparative architecture of TS/MP versions, namely TS/MP 2.0, 2.1, 2.3, 2.4, 2.5, 2.6, and 2.7.

Table 1: Comparative Architecture of TS/MP Versions

	TS/MP 2.3, 2.4, 2.5, 2.6, and 2.7	**TS/MP 2.1**	**TS/MP 2.0**	**TS/MP with NonStop Tuxedo**
Application	TS/MP Application Environment	TS/MP Application Environment	TS/MP Application Environment	TS/MP Application Environment
Management and Control	TS/MP 2.3, 2.4, 2.5, 2.6, and 2.7 (Updated PATHMON, PATHCOM and new PDMCOM processes)	TS/MP 2.1 (Existing PATHMON and PATHCOM Processes)	TS/MP 2.0 (Existing PATHMON and PATHCOM Processes)	TS/MP Tuxedo (Supervisor and TSMPCOM Processes)
Link Management	Application Cluster Services (ACS)	Application Cluster Services (ACS)	LINKMON Process	Router Process
Operating System	NonStop J, H-Series, or L-series Operating System	NonStop G-Series Operating System	NonStop J, G or H-Series Operating System	NonStop J, G or H-Series Operating System

Objects and Processes Provided by NonStop TS/MP

The elements of a PATHMON environment are provided by the NonStop TS/MP product. Information about configuring and managing these items is given in this manual.

- PATHMON— The process that defines and manages SERVER objects and—if your environment includes Pathway/iTS—TCPs, TERM objects, and PROGRAM objects. 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— The 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 OSS server programs, which reside in the OSS operating environment.
- Application Cluster Services (ACS) Subsystem Processes— These processes use Pathsend requestors to provide link management functions for applications. (Pathsend requestors are user applications that use Pathsend procedure calls to send requests to server classes.) In TS/MP, ACS subsystem core processes replace the TS/MP 2.0 LINKMON process. (Although TS/MP has a ROUT process with $ZLnn names, similar to what TS/MP 2.0 uses, the TS/MP ROUT is a redirector process that is part of the ACS subsystem. The TS/MP ROUT object is also located in the SYSnn subvolume, not in the SYSTEM subvolume.) The ACS subsystem must be configured and started (that is, $ZACS must be running) before using Pathsend requestors. For more information on ACS subsystem, see the *TS/MP 2.7 ACS Reference Manual*.
- Pathway Domain Management Interface (PDMI)— A new management interface introduced in TS/MP. PDMI is an interactive management interface with a set of commands to configure, operate, control, and collect information for multiple PATHMON processes, with replicated configurations, under a single Pathway domain. Additionally, TS/MP has enhanced PATHMON and PATHCOM processes, which now provide support for the new PDMI. For more information on PDMI, see the *TS/MP 2.7 ACS Reference Manual*.

  NOTE: PDMI is configured and managed by a new program, called PDMCOM, which provides a command-line interface. The PDMCOM program is a superset of the PATHCOM command interface, and by default, the PDMCOM object resides in $SYSTEM.SYSTEM.

Objects and Processes Provided by Pathway/iTS

The elements of a Pathway environment are provided by the Pathway/iTS product. Information about configuring and managing these items is covered in the *Pathway/iTS System Management Manual*.

- TCP objects—Terminal control processes that run screen (SCREEN COBOL) programs and coordinate communication between screen programs, input-output devices or processes, server processes, and the PATHMON process.
- TERM objects—Definitions of tasks that use screen programs to control inputoutput devices such as terminals and workstations, or input-output processes that allow users to communicate with an OLTP application.
- PROGRAM objects—Templates for creating and starting temporary TERM objects.

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 Remote Server Call (RSC) product, and the Pathway Open Environment Toolkit (POET) run-time environment.

Figure 1: A NonStop TS/MP Application With Pathway/iTS on page 17 illustrates a Pathway environment that includes the NonStop TS/MP product and the Pathway/iTS product. In this example, a remote system uses Pathsend requestors to access the server processes, with link management provided by ACS subsystem processes. A set of Pathway/iTS application terminals makes requests through the TCP. Additional elements of the application include the PATHMON process that manages the TCP, TERM, and SERVER objects, a command terminal running PATHCOM, and a management programming application.

Figure 2: NonStop TS/MP Application With Guardian and OSS Servers on page 18 shows an example of a NonStop TS/MP application configured in two operating environments. A remote system sends Pathsend requests to the ACS subsystem processes, which manages communications with the server classes. One server class is configured in the OSS operating environment. The other server class resides in the Guardian operating environment. The PATHMON process, the PATHCOM interactive interface, and a management programming application are all Guardian processes.

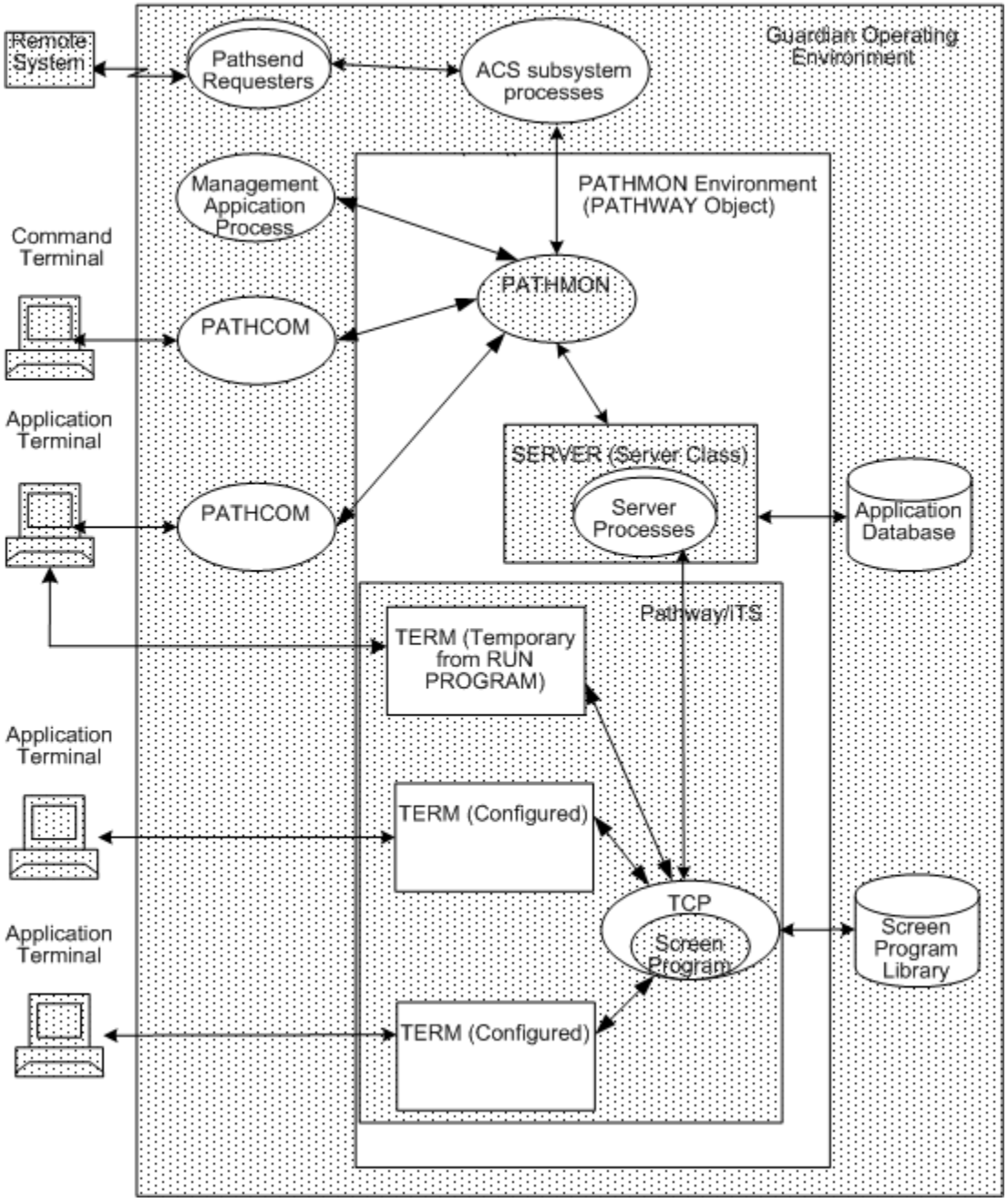

Figure 1: A NonStop TS/MP Application With Pathway/iTS

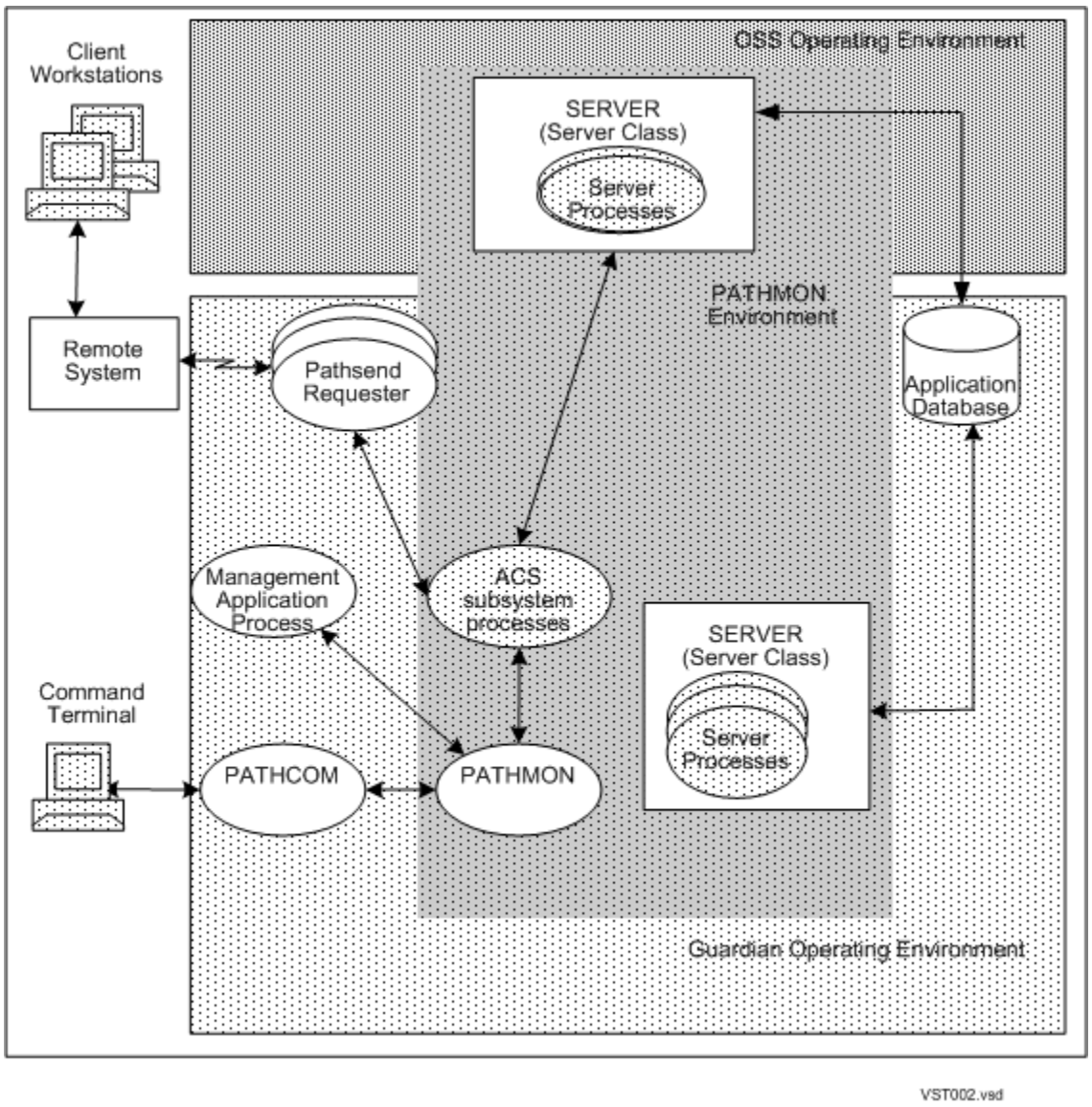

Figure 2: NonStop TS/MP Application With Guardian and OSS Servers

Distributing a Pathway or PATHMON Environment

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

One way to optimize response time for your business transactions might be to distribute your application's workload across more than one processor in a NonStop system. **Figure 3: PATHMON-Controlled Objects Distributed Over Two CPUs** on page 19 shows PATHMON-controlled objects configured on two CPUs in a system. This environment includes both NonStop TS/MP processes (PATHCOM, PATHMON process, server classes) and Pathway/iTS processes (TCPs and TERM objects).

You can distribute functions over different physical or geographical locations by configuring elements of a single PATHMON environment on various nodes in a network connected by communications lines, or by configuring two or more PATHMON environments to communicate across such a network. As an example, you can allow 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 4: NonStop TS/MP Application Distributed Over Two Nodes on page 20 illustrates a NonStop TS/MP application server class distributed across two nodes. The application checks accounts

and updates it using batch processing. The batch requestor process and a server class are configured on \NODEA while a second server class is configured on \NODEB. Both \NODEA and \NODEB are connected over communication lines. Additionally, there are two PATHMON processes (one on each node) managing the server classes at their respective nodes. The requestor process on \NODEA connects to the server class on \NODEB using the ACS subsystem process running on the same processor as that of the requestor process on \NODEA. The ACS subsystem processes on \NODEA communicate with the PATHMON process on \NODEB to find a link to the server class running on \NODEB.

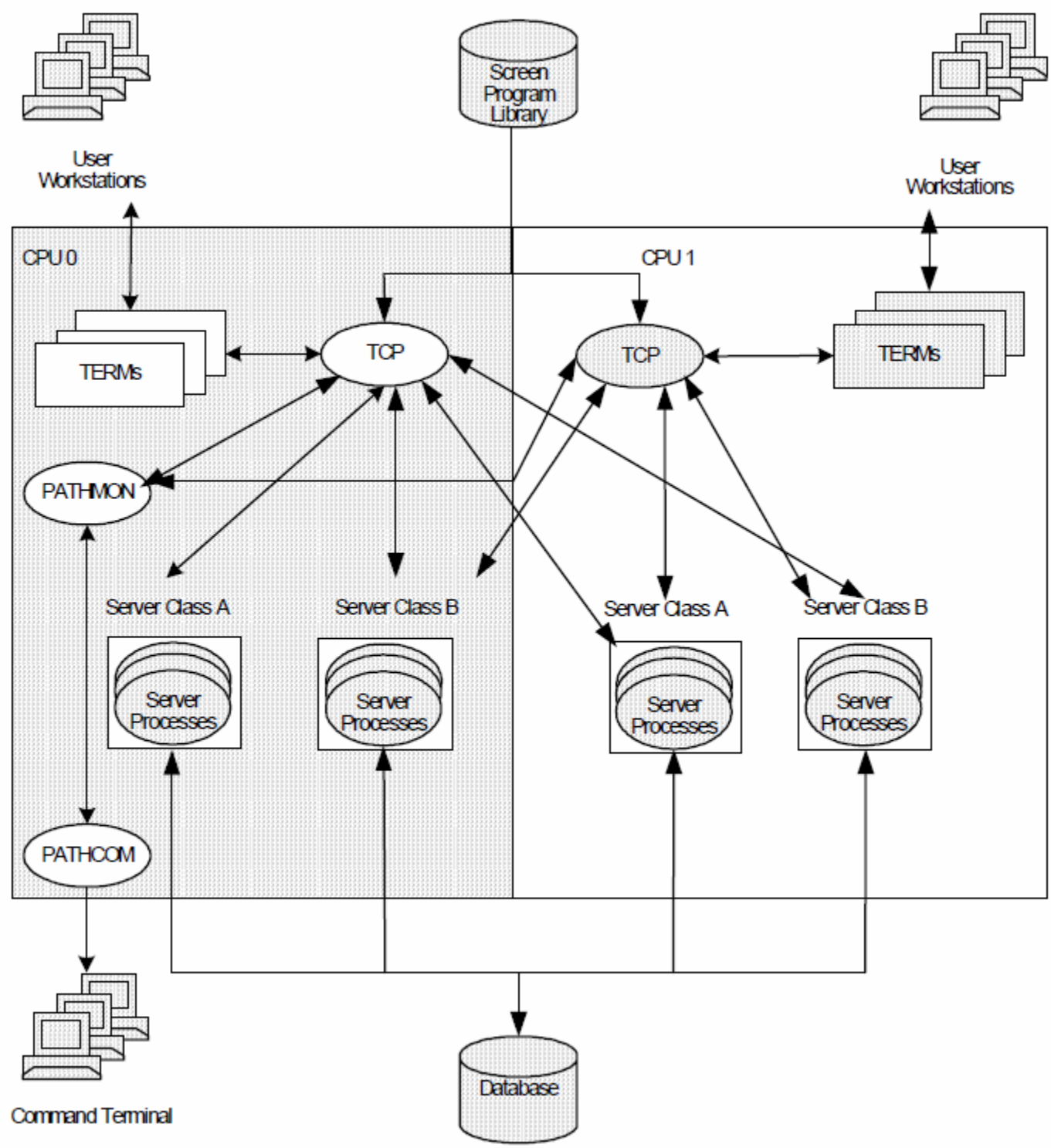

Figure 3: PATHMON-Controlled Objects Distributed Over Two CPUs

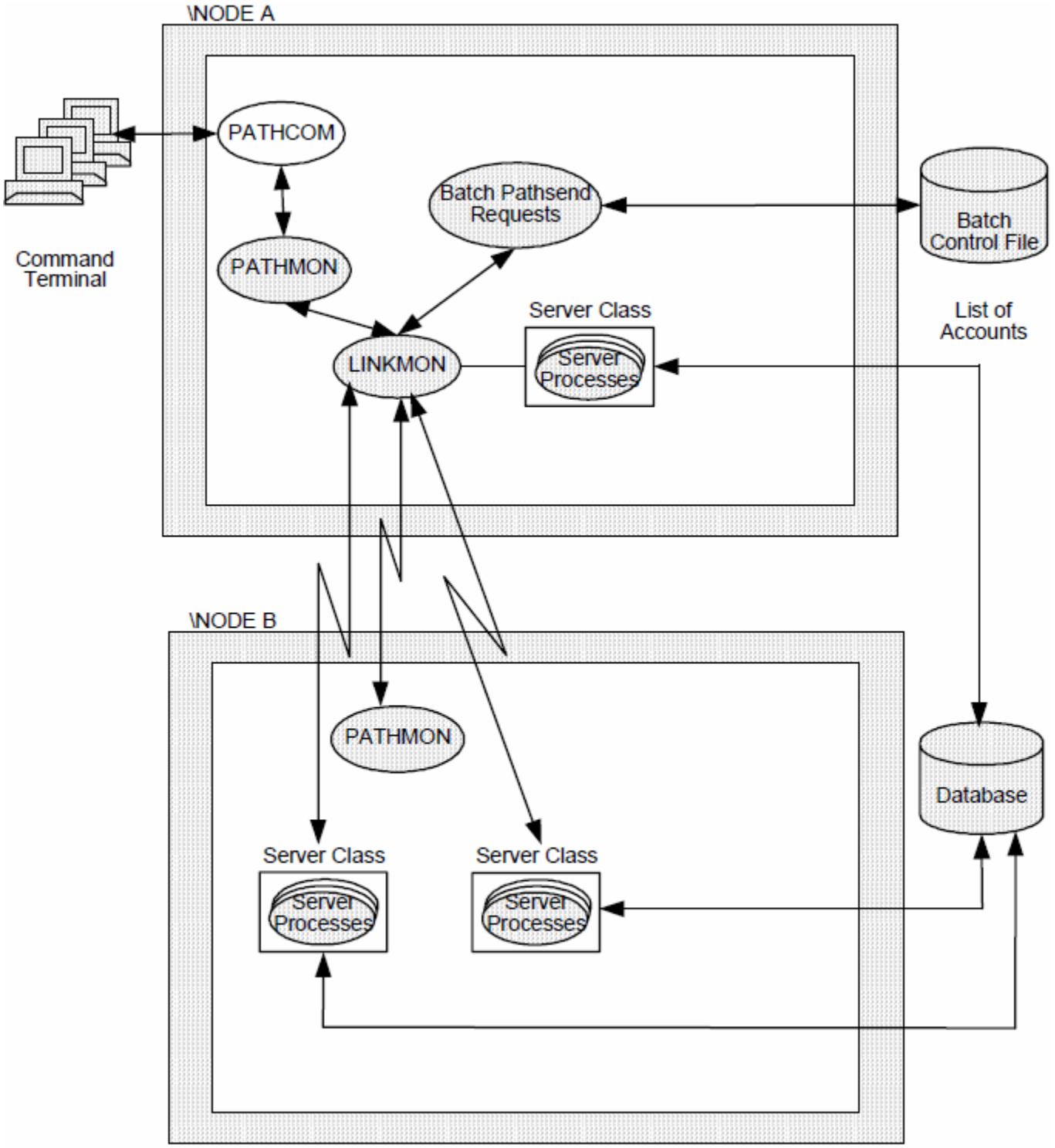

Figure 4: NonStop TS/MP Application Distributed Over Two Nodes

NonStop TS/MP Objects and Processes

This subsection contains additional detail on the transaction services objects and processes provided by the NonStop TS/MP product. Information on defining and managing these items is provided in this manual. TS/MP 2.6 and later has updated PATHMON, PATHCOM and PDMCOM processes. For more information, see the *TS/MP 2.7 ACS Reference Manual*.

PATHMON Object

The PATHMON object represents the PATHMON process. The PATHMON process and the objects it controls make up the PATHMON environment. Each PATHMON environment has only one PATHMON process, the main control process in the environment.

The PATHMON process defines, configures, and manages all the objects under its control. These objects include server objects and—if your environment includes Pathway/iTS—TCPs, TERM objects, and PROGRAM objects. The PATHMON process creates and maintains a file in which it stores the configuration information for each of these objects. When your application is running, the PATHMON process communicates with all the objects, receiving information about them and managing their activities.

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

- Keeping a record of the object definitions in the PATHMON configuration file.
- Starting SERVER processes (available only if you are running the NonStop TS/MP product).
- Starting TCPs, TERM objects, and PROGRAM objects (available only if you are running the Pathway/iTS product).
- Facilitating access between link managers such as ACS subsystem processes or TCPs, and server processes to support requestor/server communication.
- Reporting status information about PATHMON-controlled objects.
- Reporting system errors caused by invalid or unsuccessful operations.
- Shutting down all or part of a PATHMON environment by stopping individual objects.
- Automatically restarting failed objects.

PATHWAY Object

The PATHWAY object represents the entire set of objects controlled by the PATHMON process. You refer to it to set attributes affecting the entire PATHMON environment, start up and shut down the PATHMON-controlled objects, and obtain status information about the PATHMON environment. For information on the attributes that affect the PATHWAY object, see **Starting and Stopping a PATHMON Environment** on page 33.

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. In general, Guardian server processes respond to requests from SCREEN COBOL and Pathsend requestors, and OSS server processes respond to requests from OSS processes such as NonStop TUXEDO clients. If required,

however, processes in the OSS environment can be coded to send requests to Guardian server processes; and SCREEN COBOL and Pathsend requestors can send requests to OSS server processes.

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

All processes within a server class run copies of the same server program, which means that all processes within a server class provide the same set of functions. This replication of server-class function allows you to distribute the transaction workload across multiple CPUs in a system.

The number of server processes configured in a server class can vary. It is usually determined by application response-time requirements and the transaction workload. The PATHMON process can start additional server processes during peak workloads to maintain acceptable throughput.

ACS Subsystem Processes

An ACS subsystem process is a multitasking process that handles the interface to server processes from requestors other than SCREEN COBOL and TUXEDO requestors. Such requestors include Pathsend requestors; workstation clients that use RSC and, optionally, POET.

Each processor in a system has one ACS subsystem process, which starts when the processor starts. The ACS subsystem process is shared by all Pathsend requestors running in a given processor. ACS subsystem processes function as link managers, providing link management functions for Pathsend requestors in the way that the TCP and the PATHMON process provide those functions for SCREEN COBOL requestors.

NOTE:

Although ACS subsystem processes are visible to the PATHMON process, they are not entities controlled or managed by the PATHMON process. For more information about the process and Pathsend requestors, see **Managing the Pathsend Environment** on page 112.

Transaction Sources

A variety of devices and processes might interact with the PATHMON process and objects to supply required resources and services or to make requests. These include PATHCOM or a management application process (described in **System Management Tools** on page 28), as well as terminals controlled by TCP objects. In addition, these devices and processes might include:

- Personal computers and workstations (using the Remote Server Call (RSC) product).
- External TCP objects, that is, Pathway/iTS 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 ACS subsystem processes).

Figure 5: Transaction Sources on page 23 shows devices that interact with PATHMON-controlled objects, either directly or through the ACS subsystem processes.

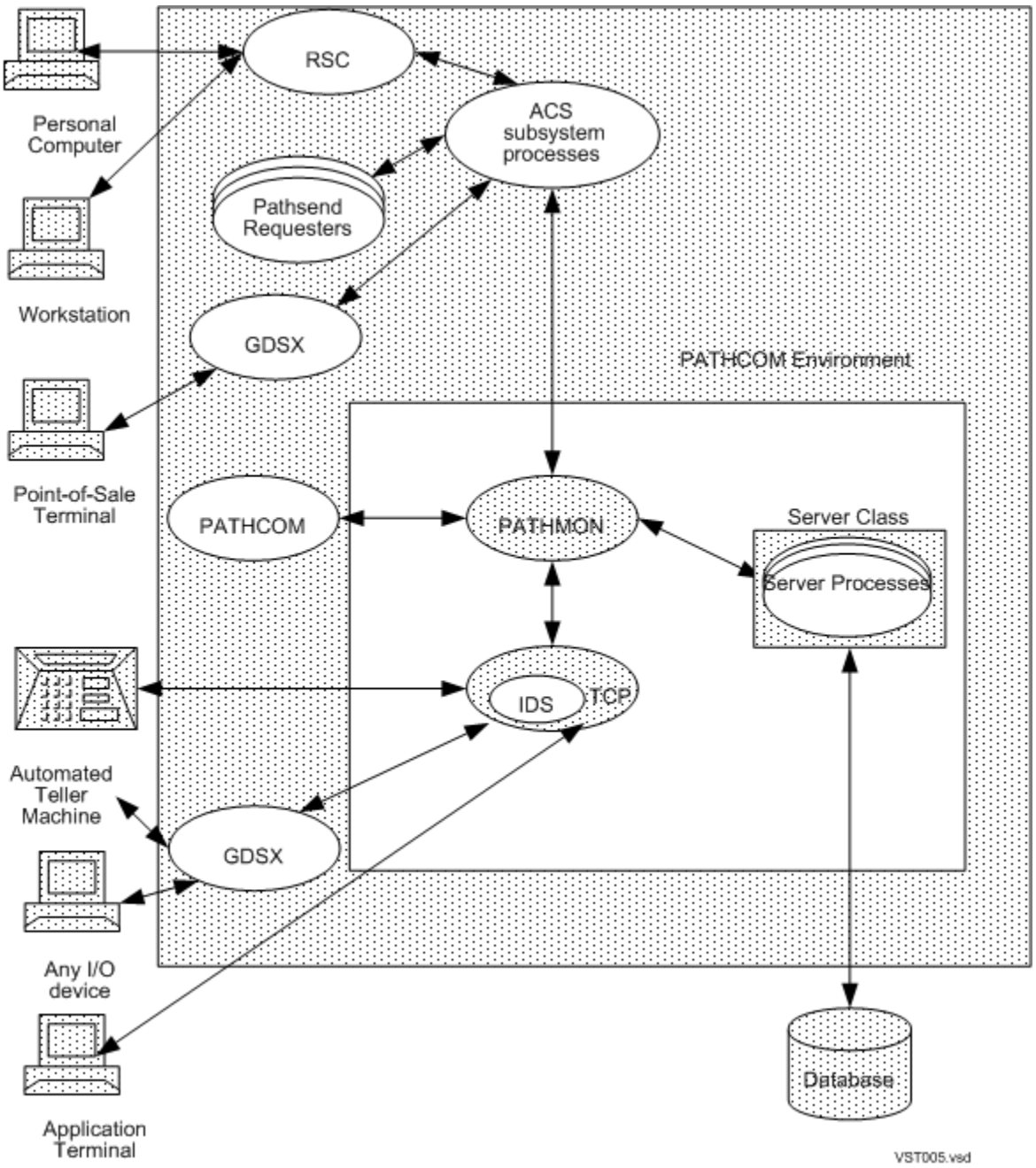

Figure 5: Transaction Sources

Personal Computers and Workstations

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

RSC allows client-server computing by supporting a variety of hardware and software configurations and communications protocols for PCs and workstations. In addition to delegating some processing normally performed by the NonStop system to the PC, RSC 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 product, see the **Remote Server Call (RSC) Installation and Management 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. The system manager who configures a PATHMON environment determines how many (if any) external TCPs can communicate with the PATHMON environment.

Remember that 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. Information about defining and managing TCPs is covered in the *Pathway/iTS System Management Manual*.

Intelligent Devices

Intelligent devices—such as automated teller machines (ATMs), workstations, and barcode readers—can access server classes through SCREEN COBOL requestor programs and the TCP, a process supported under the Pathway/iTS product.

This access route is possible when the intelligent device support (IDS) facility, which is part of the TCP, is used. 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.

SNA Devices

Requests from SNA devices can be handled by SNAX High-Level Support (SNAX/HLS), which allows NonStop application programs to communicate with SNA devices and host software. Using SNAX/HLS, NonStop 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 HPE 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 ACS subsystem processes, or (under Pathway/iTS) a TCP.

GDSX is a complex product. Before choosing to use GDSX, determine whether the ACS subsystem processes can manage the requests alone. If your environment includes Pathway/iTS, investigate whether the PATHMON process and TCP can handle your application requirements. If the specified data communications protocols are not supported by existing Pathsend requestors or (under Pathway/iTS) a TCP, you might decide to use a GDSX process as a front end to the application.

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 server processes. For more information about the GDSX product, see the *Extended General Device Support (GDSX) Manual*.

Pathsend Processes

Pathsend processes (also called Pathsend requestors) are user applications that access server classes using Pathsend procedure calls. These procedure calls are part of the Guardian procedure library. RSC and POET workstation clients are examples of applications that communicate with server processes

indirectly by means of Pathsend requestors created by RSC on behalf of the clients. The ACS subsystem processes manage communications between Pathsend processes and server classes. System management issues related to Pathsend are described in **Managing the Pathsend Environment** on page 112 of this manual. (For information about how to write Pathsend programs, see the *TS/MP 2.5 Pathsend and Server Programming Manual*).

Figure 6: Requestor Access to Server Classes on page 25 shows a PATHMON environment with Pathsend requestors. Link management functions for Pathsend requestors are provided by the ACS subsystem processes $ZL01 and $ZL02. The ACS subsystem processes service requests only from Pathsend processes executing in its processor. $PMB is the PATHMON process controlling the pathway environment.

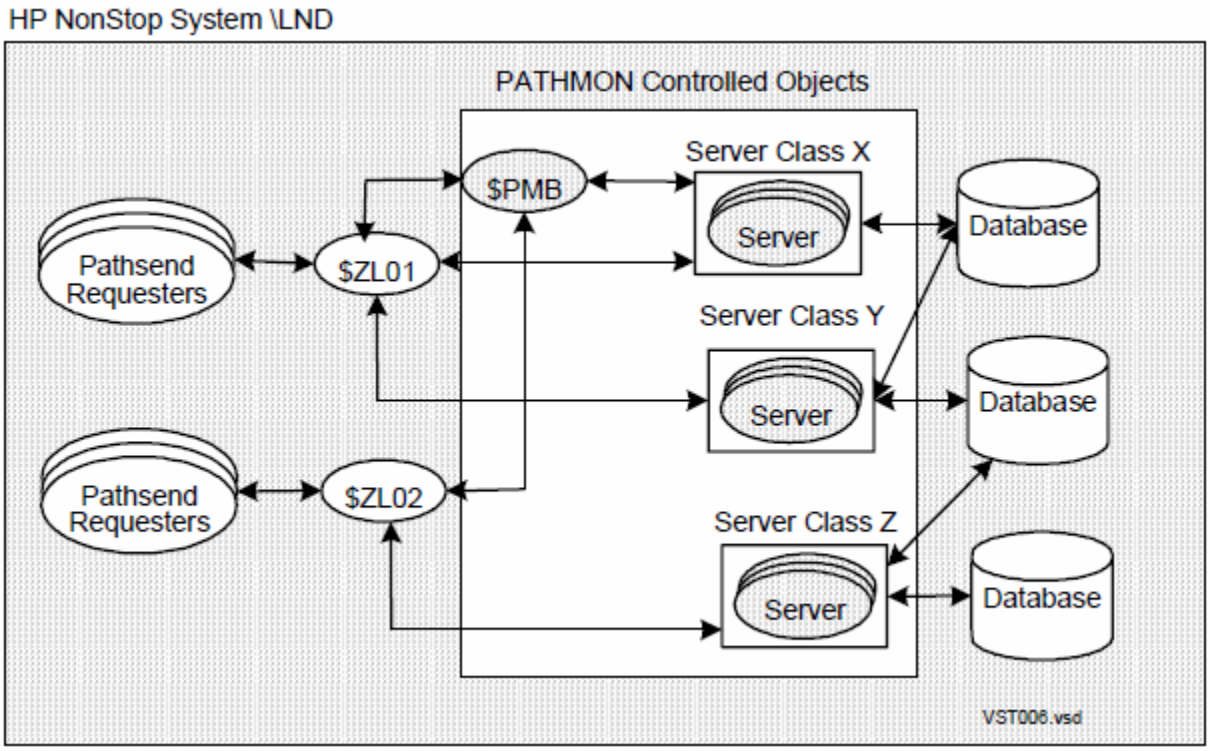

Figure 6: Requestor Access to Server Classes

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 tasks shown in **Figure 7: System Management Tasks** on page 26. You can perform the required tasks using PATHCOM or by writing a management application. For details, see **System Management Tools** on page 28.

NOTE: System Management tasks can be performed using PDMCOM or PATHCOM. When using TS/MP 2.6 or later, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

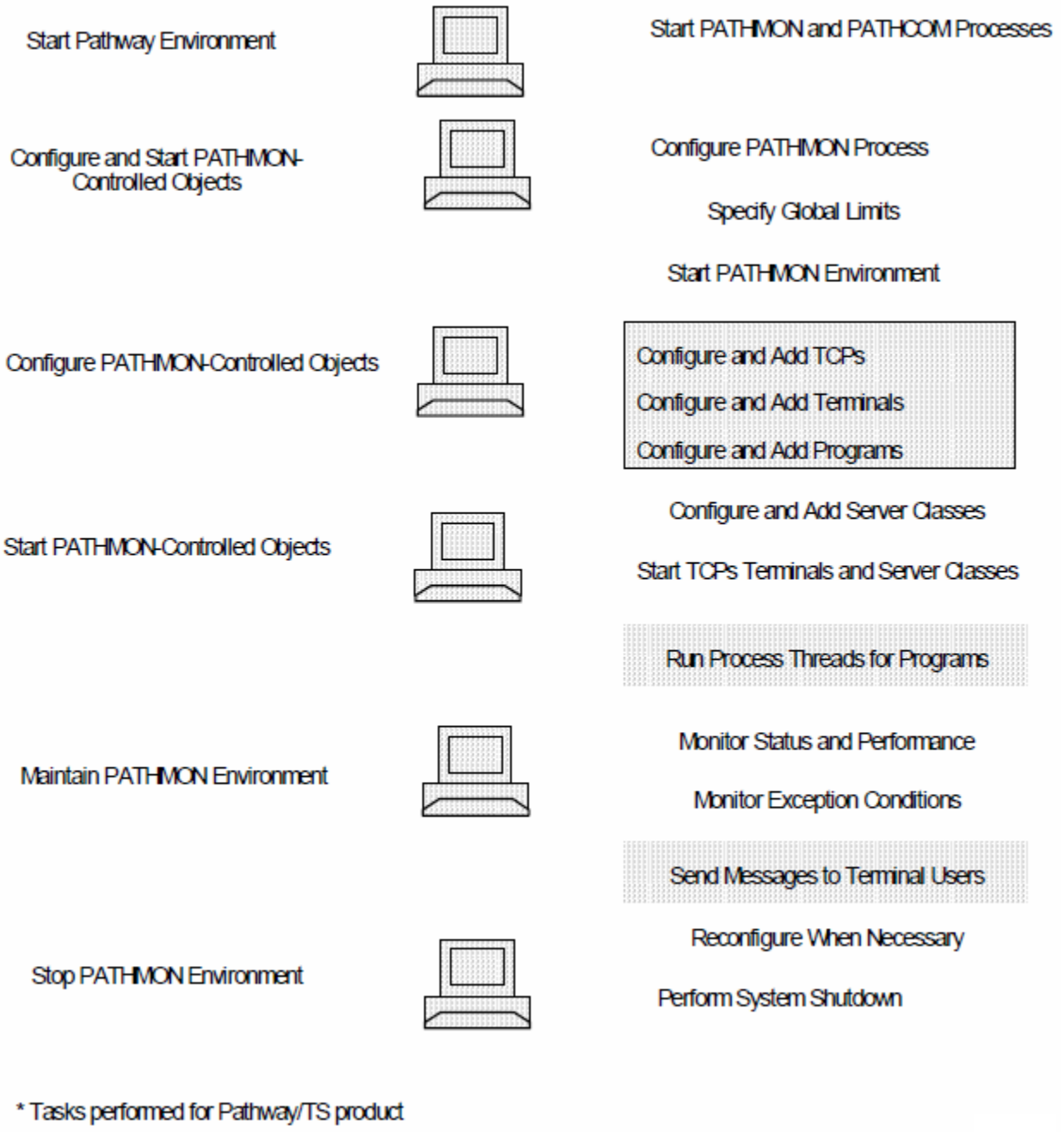

Figure 7: System Management Tasks

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.
- Start the PATHMON environment by using the `START 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.

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 on a previous session and do not need to reconfigure 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 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 7: System Management Tasks** on page 26 that are specific to Pathway/iTS—those that define, configure, and manage the TCP, TERM, and PROGRAM objects—are described in detail in the *Pathway/iTS System Management Manual*.

For complete information about the PATHCOM commands that configure and control the PATHMON environment and PATHMON-controlled objects, including syntax, usage rules, and error messages, see Sections 8 through 16. For a summary of these commands, see **Appendix A, Syntax Summary**.

System Management Tools

You can choose one of two ways to configure and control PATHMON-controlled objects in a PATHMON environment:

- Interactively, by entering PATHCOM commands at a terminal
- Programmatically, by writing a management application that uses a Pathway management programming interface

Both of these methods allow you to send commands and instructions to the PATHMON process, the process that monitors your environment and directs the activities, as shown in **Figure 8: System Management Interfaces** on page 28.

NOTE: When using TS/MP 2.6 or later, it is recommended to use PDMCOM (instead of PATHCOM) to configure and control PATHMON-controlled objects in a PATHMON environment and for domain management, because PDMCOM can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

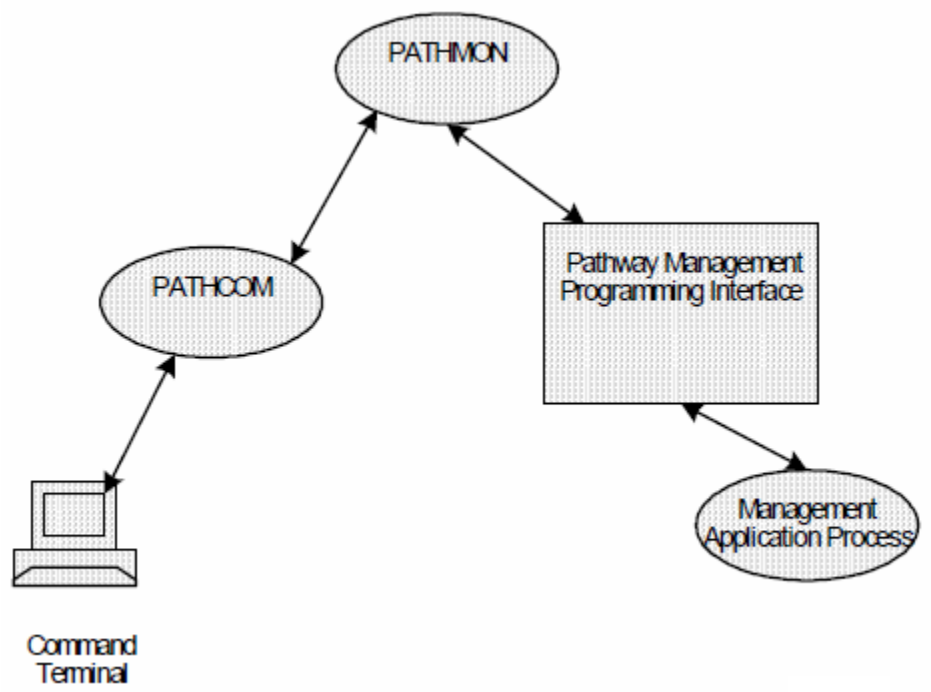

Figure 8: System Management Interfaces

PATHCOM is an interactive, text-based interface; for example, an operator using PATHCOM can easily start and stop objects or obtain status information about objects. A management programming interface is command-driven and more toward logical processes. A management programming interface can simplify the automation of tasks associated with starting, maintaining, and stopping an environment or maintaining a state.

Whether you use the PATHCOM interactive interface or the management programming interface to manage your PATHMON environment might depend on the complexity and frequency of your system management tasks. The more complex or repetitive your system management tasks, the more advantageous it is to use a management programming interface.

Your choice might also depend on the availability of resources to develop a management application, which is a sophisticated programming assignment, or on the availability of other products, such as

NonStop NET/MASTER. (NonStop NET/MASTER belongs to the set of Hewlett Packard Enterprise management products known collectively as Distributed Systems Management (DSM), which is described in **Distributed Systems Management** section available in **Management Programming Interface**.)

NOTE: PDMCOM does not support SPI commands.

PATHCOM Interactive Interface

PATHCOM provides commands for defining and managing a PATHMON environment. PATHCOM also provides online help and error messages.

PATHCOM functions, usage, and command syntax are given in **Overview of PATHCOM** on page 130 through **SERVER Commands** on page 194 of this manual. A summary of PATHCOM command syntax is displayed in **Syntax Summary** on page 351.

Using PATHCOM, which consists of sets of object-related commands, you can interactively define and manage all PATHMON-controlled objects. PATHCOM runs some of these object-related commands; other commands are processed by the PATHMON process and might involve one or more TCP or ACS subsystem processes in the execution. For example:

- You define PATHMON-controlled object configurations, start and stop objects, and obtain status information about objects by entering commands that PATHCOM passes directly to the PATHMON process.
- You request server-class performance statistics by issuing a `STATS SERVER` command. PATHCOM passes the request directly to the PATHMON process, which passes the request to the link managers, such as the TCPs and ACS subsystem processes. The link managers run the command by gathering and returning statistical information about their links to the specified server class.

PATHCOM allows you to enter commands interactively through a terminal or from a command file or TACL script. For more information about using command files and TACL scripts, see **Starting and Stopping a PATHMON Environment** on page 33.

Management Programming Interface

A management programming interface is a token-oriented interface based on the Subsystem Programmatic Interface (SPI) and the Event Management Service (EMS)— part of HPE's Distributed System Management (DSM) product. (See **Distributed Systems Management** section below.)

The management programming interface acts as the interface between a management application process—(a requestor that you write)—and the PATHMON process, which acts as the server.

As shown in **Figure 9: Management Programming Environments** on page 30, a management application in the operations environment uses the management programming interface to monitor and control objects in the PATHMON environment. The management application sends commands and inquiries to processes in the PATHMON environment through the management programming interface. The processes send responses or event messages to the management application using the same interface.

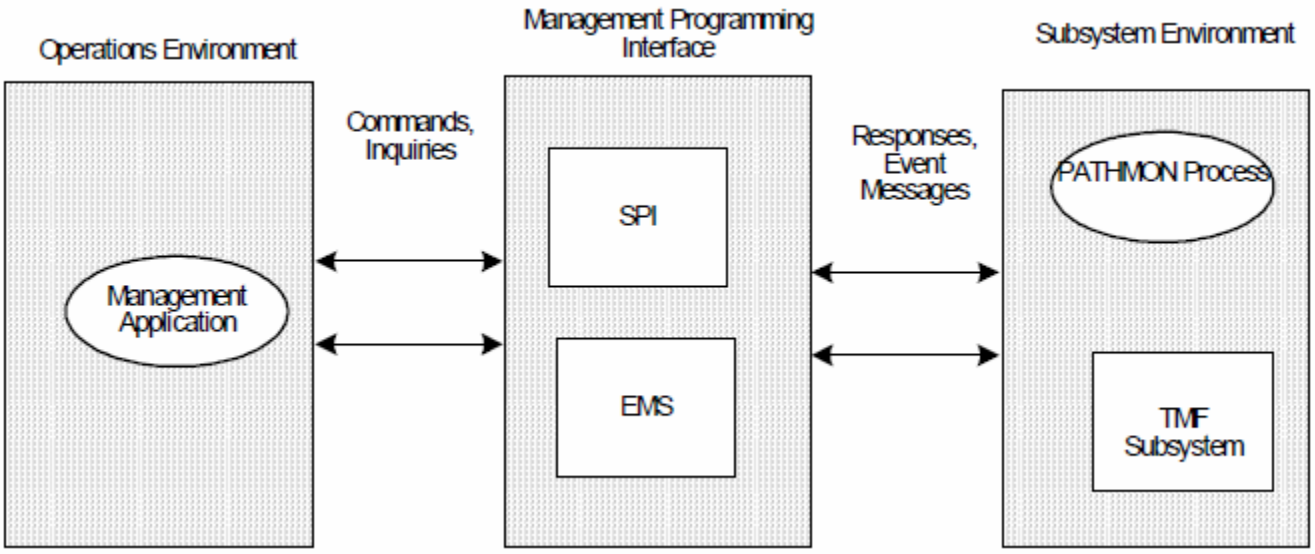

Figure 9: Management Programming Environments

Distributed Systems Management

The Distributed Systems Management (DSM) set of products is designed to monitor and control every element of a distributed network, including PATHMON-controlled objects in a Pathway environment. DSM incorporates a three-part architecture:

- Operations environment

  The operations environment includes your management application, which resides in the same run-time environment as standard Guardian applications. The application can be written in languages such as COBOL, C, TAL, and TACL.

- Management programming interface

  The management programming interface provides the standard set of procedures, definitions, and message formats required to send messages between your management application in the operations environment and the PATHMON process in the Pathway environment. The management programming interface includes SPI and the Event Management Service (EMS), another DSM facility. EMS provides collection, logging, and distribution of event messages, which are noteworthy occurrences such as state changes in objects or component failures. The PATHMON process reports events to an EMS collector process, which logs the events in an event log. An EMS distributor process reads the log and distributes the events to all appropriate management applications.

- Subsystem environment

  The subsystem environment includes the PATHMON process and objects. The Transaction Management Facility (TMF), which provides transaction processing protection, also resides in this environment.

For more information about the Distributed Systems Management set of products, see the **Introduction to Distributed Systems Management**.

Advantages of the Management Programming Interface

By writing your own management application that uses the management programming interface, you can automate tasks and add special functions not available within PATHCOM.

A management application issues the system management commands and provides a command interface, formatting, and error handling. A management application can also interact with subsystems outside of the Pathway environment. Thus, the management application can provide a focal point for many different kinds of functions.

Your application might ask the PATHMON process to examine the status of a device, and then use other subsystems to gain more information about the device, shut it down, and later restart it. For example, you might write management applications that:

- Start PATHMON-controlled SERVER objects automatically at a specific time of day. (This application communicates exclusively with the PATHMON process.)
- Monitor the status of TERM objects and, if a device is not running, contact the data communications subsystems to obtain information about the communications line to which the device is connected. (This application interacts with both the PATHMON process and the data communications software.)
- Receive event messages for PATHMON-controlled objects and issue commands to find out more information about why the event is being reported. (This application communicates with both the PATHMON process and the Event Management Service (EMS), supplied as part of DSM for collecting and reporting events.)

Management applications that use the management programming interface can improve systems availability by reducing the chance of human error, which often occurs during routine system management. Also, by automating repetitive system management tasks, management applications can free operators to concentrate on more critical or creative system management activities.

Other System Management Tools

You get optimum performance from a well-balanced NonStop system. These tools allow you to analyze NonStop system performance.

- The Measure product allows you to balance and tune NonStop systems. Using the Measure product, you can configure measurements, collect performance data, display and plot data, and so on.

  For more information about using Measure, see the *Measure User's Guide and the Measure Reference Manual*.

- The HPE Tandem Performance Data Collector (TPDC) is a host-based performance collection and relationship product. TPDC integrates and normalizes data from Measure and other sources and creates a single consolidated file for subsequent reporting and analysis.

  TPDC is bundled with the System Performance Analysis and Measurement (SPAM) set of products.

- The ViewPoint operations console application lets you interact simultaneously with multiple NonStop products, including PATHCOM. Using the ViewPoint application, you can control an integrated NonStop system, including many products, from one location. For more details about the ViewPoint application, see the *ViewPoint Manual*.
- The EMS FastStart product enhances development of applications under DSM. EMS FastStart provides a simple, cost-effective way for programmers to develop and test event messages. For more information, see the *EMS FastStart Manual*.
- Service provider is a customer-support program designed to provide a wide range of service products to assist customers during all phases of system planning, design, implementation, and production. The services are designed as modules so that you can choose the specific areas in which you need assistance. For information about these services, contact your Hewlett Packard Enterprise representative.

Using these tools, you can make informed decisions about configurations in your PATHMON environment.

Summary of PATHCOM Commands

This list describes some of the PATHCOM commands you use to manage PATHMON-controlled objects:

- `START` command starts the PATHMON-controlled objects in a PATHMON environment.
- `SET` command defines configuration values for the object type.
- `SHOW` command displays the current values for an object.
- `ADD` command adds a PATHMON-controlled object.
- `INFO` command displays values for an object as recorded in the PATHMON configuration file.
- `STATUS` command displays the status of an object (whether it is running or stopped, for example).
- `STATS` command displays resource usage and system performance statistics.
- `ALTER` command changes the attributes of a previously defined object.
- `STOP` command stops individual objects, for example, SERVER objects.
- `SHUTDOWN2` command stops all PATHMON-controlled objects.

For complete information about the PATHCOM commands that configure and control the PATHMON environment and PATHMON-controlled objects, including syntax, usage rules, and error messages, see Sections 8 through 16. For a summary of these commands, see **Syntax Summary** on page 351.

Starting and Stopping a PATHMON Environment

Steps to Starting and Configuring a PATHMON Environment

This section describes how to start, restart, and shut down a PATHMON environment. Before you actually start a PATHMON environment, you start the PATHMON process. Multiple PATHMON environments can run on a given node; however, each PATHMON environment includes only one PATHMON process. The PATHMON process enforces the limits you set for the environment and monitors the operation of all objects (for example, SERVER objects) under its control. Note that although this section gives an overview of the steps involved in setting limits for the environment, for complete instructions on configuring your PATHMON environment, see **Configuring Objects in a PATHMON Environment** on page 50.

After you start the PATHMON process, you start PATHCOM, the interactive interface that allows you to communicate with the PATHMON process. Now you are ready to enter the commands that define the limits for your PATHMON environment. At system startup, the PATHMON process uses these limits to create the PATHMON configuration file.

NOTE:

The PATHMON environment includes the PATHMON process and the objects it controls: SERVER objects and-if your environment includes Pathway/iTS-TCPs, TERM objects, and, PROGRAM objects.

How the PATHMON Process Builds the Configuration File

The PATHMON configuration file, whose default name is PATHCTL, stores configuration information about your Pathway environment. Configuration information is all the data about environment limits and object attributes that is stored as a result of the `SET`, `ADD`, `ALTER`, `DELETE`, `CMDCWD`, `CMDVOL`, or `CONTROL` commands.

When you start a new system, the PATHMON process creates the `PATHCTL` file using the limits that you specify in the `SET PATHWAY` commands prior to startup. The PATHMON process reserves enough disk space to store configuration definitions for all objects. For example, if you specify a maximum of five server classes, the PATHMON process builds the configuration file with space for five server class configurations.

When you define and add objects to your PATHMON environment, the PATHMON process fills in the appropriate areas in the PATHMON configuration file with the object definitions. During system operation, the PATHMON process updates the configuration file as object definitions are added, altered, or deleted. Whenever you cool start a system, the PATHMON process uses the information stored in this file to create the operating environment.

Assigning the PATHMON Configuration File

The PATHMON configuration file is always automatically created on the node where the PATHMON process is running. Also, unless you specify otherwise, the PATHMON process operates as if the PATHMON configuration file is located in the default volume and subvolume, under the name PATHCTL.

You can, however, assign unique names to the PATHCTL file by using the ASSIGN command prior to starting the PATHMON process. For example, this ASSIGN command, entered at the TACL prompt, specifies a configuration file named PMCNTL on subvolume B of volume $A:

```
38> ASSIGN PATHCTL, $A.B.PMCNTL
```

Multiple configuration files with the same name−PATHCTL, for example−can exist on the same NonStop system; however, only one file with a given name can exist on a given volume and subvolume. If you plan to start more than one PATHMON process from a given subvolume, you must assign a unique name to each configuration file; otherwise, a PATHMON process might attempt to use the wrong configuration file, resulting in this error:

```
ERROR *1017* PATHCTL FILE (file-error-num)
```

The PATHMON process saves the configuration values you specify in memory, as shown in **Figure 10: Configuring the PATHMON Process** on page 34. At startup, these values are written to the PATHMON configuration file. (For more information, see **Configuring the PATHMON Process** on page 40.)

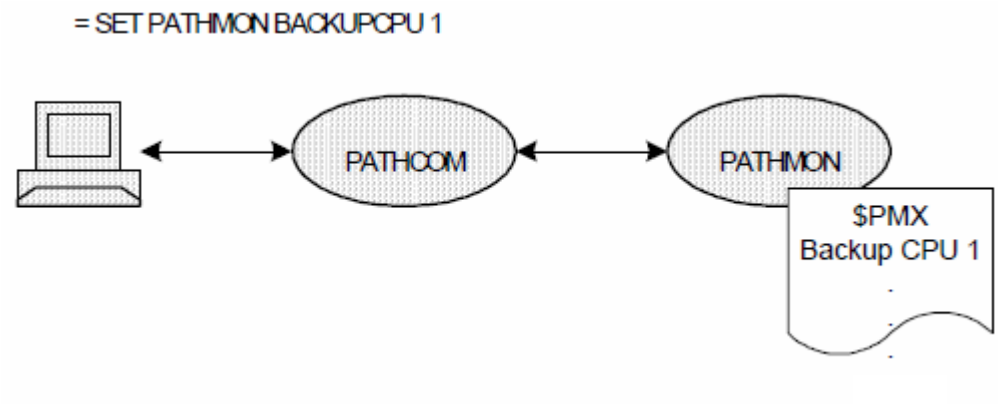

Figure 10: Configuring the PATHMON Process

Starting a PATHMON Environment

Starting a PATHMON environment consists of these steps, which are summarized in this list and explained in detail throughout this section:

1. Start the PATHMON process.

   Assign a process name and start the PATHMON process that controls your PATHMON environment.

2. Start PATHCOM.

   Start a PATHCOM process that communicates with the PATHMON process. From this point on, all your interactions with the PATHMON environment take place through PATHCOM.

3. Configure the PATHMON environment.

   Use SET PATHWAY commands to specify global parameters and limits for the overall configuration.

   Use SET PATHMON commands to specify a backup processor and request error dumping.

   One of your options is to configure all processes and devices in the Pathway environment as node-independent, making it easier to move your database to a new system. Node names for node-independent process and device names default to the node where the PATHMON process is currently running.

4. Start the Pathway environment.

Once these four steps are completed, you can use PATHCOM to configure the SERVER objects controlled by the PATHMON process. Object configuration is described in **Configuring Objects in a PATHMON Environment** on page 50. For instructions on configuring TCPs and TERM and PROGRAM objects, see the *Pathway/iTS System Management Manual*.

NOTE: For starting the PATHMON environment in TS/MP 2.6 or later, PDMCOM or PATHCOM can be used. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

Starting the PATHMON Process

To start a new PATHMON process, name the PATHMON process in an implicit `RUN` command to TACL. For example, this command starts a PATHMON process named $PMX and specifies that the process will run on processor 3:

```
> PATHMON /NAME $PMX, CPU 3, NOWAIT/
```

You can enter this command from your terminal, as shown in the example. Alternatively, you can create a command file that contains this command (as well as others) and then request execution of the commands in that file. In either case, the operating system responds to your command by creating and starting the primary PATHMON process, as shown in **Figure 11: Starting the PATHMON Process** on page 35.

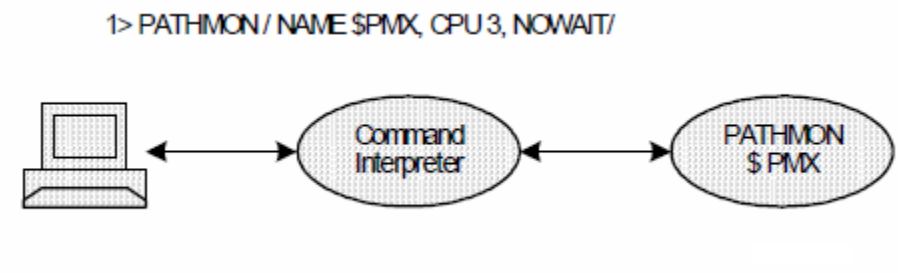

Figure 11: Starting the PATHMON Process

Always provide a name for your PATHMON process. You use this name to identify the PATHMON process when you communicate with the PATHMON process through PATHCOM. (If you do not supply a name, the system gives the PATHMON process a default name, $X984. Hewlett Packard Enterprise recommends that you provide your own PATHMON name, which is easier to remember.)

The name you choose for the PATHMON process must start with a dollar sign ($) followed by one to five alphanumeric characters, or if the PATHMON name is used across a network, then one to four alphanumeric characters. The first alphanumeric character must be a letter.

CAUTION:

If multiple Pathway environments are configured on a node, do not assign the name $PM to any PATHMON process on that node. Your commands might be applied to the wrong PATHMON environment.

$PM is the default name used when someone opens communication with PATHCOM without specifying a PATHMON name. Suppose that your PATHMON environment runs under the PATHMON named $PMX, but you forget to specify this name. PATHCOM attempts to direct subsequent commands to a PATHMON process named $PM; if a PATHMON process with this name exists, your commands are applied to the wrong PATHMON environment.

RUN Options

In addition to the PATHMON name, two other parameters, *processor* and *NOWAIT*, are generally specified in the `RUN` command. These parameters are `Guardian RUN` options that take effect when the PATHMON process begins execution.

The processor attribute indicates the processor in which the primary member of your PATHMON process pair runs in a multiprocessor environment. You can select any processor for this purpose. If you do not specify a processor, the system selects a processor at random. Using the default value can result in an error if the PATHMON process later attempts to start the backup process in the same processor. In **Figure 11: Starting the PATHMON Process** on page 35, the primary PATHMON process runs in processor 3.

In a single-processor environment, the PATHMON process always runs as a single process in the only processor available, whether or not you specify the processor attribute.

The *NOWAIT* parameter requests the `TACL` command interpreter to run the PATHMON process under the `NOWAIT RUN` option. You specify this option so that the PATHMON process runs as a background process. TACL regains control of the terminal as soon as the PATHMON process is created.

In addition to the processor and *NOWAIT* parameters, you can specify a log file in which your PATHMON records errors and changes in object status, as shown in this example:

```
5> PATHMON / NAME $PMX, CPU 3, NOWAIT, OUT LOGPMON/
```

For more information about logging status and error information, see **Maintaining a PATHMON Environment** on page 81.

If you do not specify these `RUN` options, the operating system assigns default values for them. For instance, the operating system assigns your terminal as the default log file device.

For more information about these and other `RUN` options, see the *TACL Reference Manual*.

Starting and Using PATHCOM

After a PATHMON process is started, you can start a PATHCOM process to communicate with the PATHMON process by issuing an implicit `RUN` command to TACL.

For example, this command starts a PATHCOM process that communicates with the PATHMON process named $PMX:

```
> PATHCOM $PMX
```

You can communicate with a remote PATHMON process if, for example, you need to configure a PATHMON environment on a remote system. If you want to communicate with a PATHMON process on another node, be sure to use a fully qualified PATHMON process name in your command, for example:

```
> PATHCOM \NYC.$PM16
```

You can enter this command from your terminal, as shown in the example. Alternatively, you can create a command file that contains this command (as well as others) and then request execution of the commands in that file. these examples show the interactive use of PATHCOM.

The operating system responds to your command by creating your PATHCOM process and establishing communication with the PATHMON process, as shown in **Figure 12: Starting PATHCOM** on page 36:

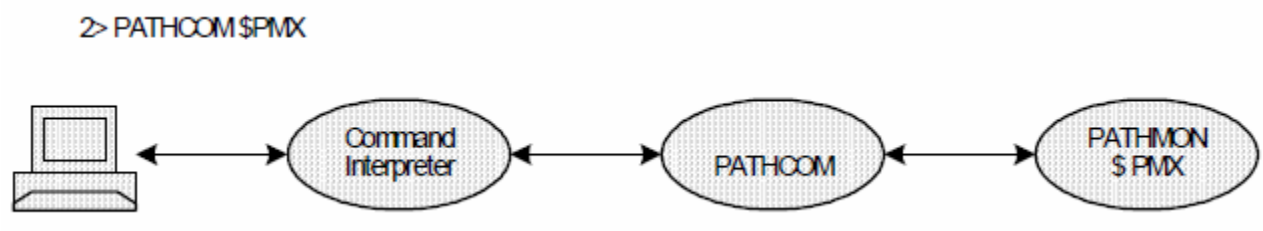

Figure 12: Starting PATHCOM

When PATHCOM starts, it opens communication with your PATHMON process, displays the PATHCOM banner message on your screen, and presents an equal sign as a prompt for your first PATHCOM command. In this example, a PATHCOM process named $X364 starts a PATHMON process named $PMX:

```
8> PATHCOM $PMX
$X364: PATHCOM - T0845H01 - (01AUG08)
(C)1980 Tandem(C)2005 Hewlett Packard Development Company, L.P.
 =
```

From this point on, you communicate with the PATHMON process only through PATHCOM, as shown by **Figure 13: Communicating With the PATHMON Process Through PATHCOM** on page 37. You are ready to enter the PATHCOM commands that request the remainder of your system management tasks. When you enter your first PATHCOM command, PATHCOM runs it and prompts you for another command.

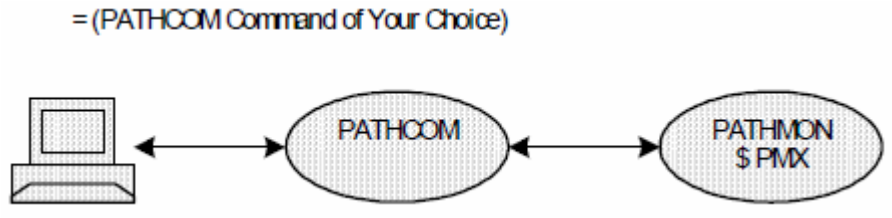

Figure 13: Communicating With the PATHMON Process Through PATHCOM

Running PATHCOM interactively is most useful when you plan to enter only a few commands. It is a good way to monitor the status of objects, dynamically start and stop objects, run a program, and send messages to user terminal operators about changing conditions. It is generally easier and faster, however, to set up lengthy configurations non-interactively.

PATHCOM RUN Options

The command to run PATHCOM, like the command to run the PATHMON process, allows you to select several `RUN` options. For instance, you can select the processor in which PATHCOM runs or specifies a device other than your terminal as the destination of output from PATHCOM. If you do not specify these options, the operating system supplies standard default values. For details about the default values for the `TACL RUN` options, see the *TACL Reference Manual*.

Getting HELP for PATHCOM Syntax and Errors

You can display a complete list of all PATHCOM commands available by entering this PATHCOM command:

```
= HELP COMMANDS
```

You can request syntax assistance from PATHCOM. This help is particularly valuable when you are entering commands interactively and would like online information about their syntax. For example, to request syntax information for the `START` command (which starts a PATHMON-controlled object), enter:

```
= HELP START
```

In response, PATHCOM displays:

```
START { <entity-list> }
      { PATHWAY <pw-starttype> [ ! ] }
      { <scname> , PROCESS <process-name> }
      { <termname-list> [ , INITIAL <puname> ] }
```

You can request information about specific command parameters, too. For instance, to request information about the *<pw-starttype>* parameter of the START command, enter:

```
= HELP <pw-starttype>
```

In response, PATHCOM displays:

```
<pw-starttype> ::=
    COLD
    COOL
```

Finally, you can also request help in interpreting message numbers that appear on object status reports from the PATHMON process. For example, if you encounter message number 1002 in a TERM status report, you can determine the meaning of that number by entering:

```
= HELP 1002
```

In response, PATHCOM displays:

```
PATHMON STATUS - *1002* ABORTED
```

Using PATHCOM in Noninteractive Mode

If you are specifying lengthy configurations, you probably want to create a command file or a TACL routine that contains your PATHCOM commands. This method permits PATHCOM to read and act upon the commands very quickly.

You enter your PATHCOM commands into the command file before starting PATHCOM. When you are ready to start PATHCOM, you specify the name of the file that you want PATHCOM to read. A command file can be an IN file or an Command file: the only difference lies in how you run the file.

IN Command

You specify the IN command when you start PATHCOM. For example, this command starts PATHCOM and directs PATHCOM to read commands from a file named PWCMD and list them on the device $S.#LP:

```
11> PATHCOM /IN PWCMD, OUT $S.#LP, CPU 1, NOWAIT/ $PMX
```

When PATHCOM runs, it reads the commands and runs them in sequence. When it encounters an end of file or an EXIT command, PATHCOM terminates.

OBEY Command

You specify the OBEY command from within PATHCOM. For example, this command directs PATHCOM to read commands from a command file named PWCMD:

```
8> PATHCOM $PMX
$X364: PATHCOM - T0845H01 - (01AUG08)
(C)1980 Tandem(C)2005 Hewlett Packard Development Company, L.P.
= OBEY PWCMD
```

TACL Routines

The TACL INLINE facility allows you to incorporate PATHCOM commands into a TACL routine to manage your PATHMON environment. This example shows a few lines from a TACL routine that starts server objects in the PATHMON configuration file:

```
        .
        .
pathcom/inline,cpu 1,pri 190 / %1% == assume local pmon
+ errors 999
[#loop |do|
#set count [#compute [count] + 1 ]
```

```
+ start server srvr-S[count]
|until| [count] >= 50
]
        .
        .
```

For more information about TACL and using TACL routines, see the *TACL Reference Manual* and the *TACL Programming Guide*.

Using DEFINEs

You can use DEFINEs to specify names for the files that PATHCOM uses directly as command files. For example:

```
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/ $PMX
```

In the preceding example, PATHCOM reads commands from the command file specified by the DEFINE =CMD-FILE ($DATA.PW.CONFIG) and lists them on the device specified by the DEFINE =OUT-FILE ($S).

For detailed information about DEFINEs, see the *TACL Reference Manual* and the *Guardian User's Guide*.

Configuring Global Parameters

After you have named your PATHMON process and opened communication with PATHCOM, you are ready to configure global parameters using the `SET PATHWAY` command.

The `SET PATHWAY` command allows you to define the limits for your overall configuration and to specify node-independent, owner, and security attributes. These tasks are briefly described in this paragraphs. For more information, see **Configuring Objects in a PATHMON Environment** on page 50.

Specifying Limits

Limits are global values that determine the maximum number of objects of each type that 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:

- 5 server classes
- 25 server processes
- 5 server STARTUP messages
- 7 server ASSIGN messages
- 5 server PARAM messages

To establish these limits, you enter these commands:

```
= SET PATHWAY MAXSERVERCLASSES 5
= SET PATHWAY MAXSERVERPROCESSES 25
= SET PATHWAY MAXSTARTUPS 5
= SET PATHWAY MAXASSIGNS 7
= SET PATHWAY MAXPARAMS 5
```

The PATHMON process saves the configuration values that you specify in memory, as shown in **Figure 14: Specifying PATHMON Environment Limits** on page 40, until you specify the `START` command, at which time the PATHMON process writes the values to the PATHMON configuration file. For the syntax of

all SET PATHWAY commands, including descriptions of all required parameters, see **PATHMON Environment Control Commands** on page 159.

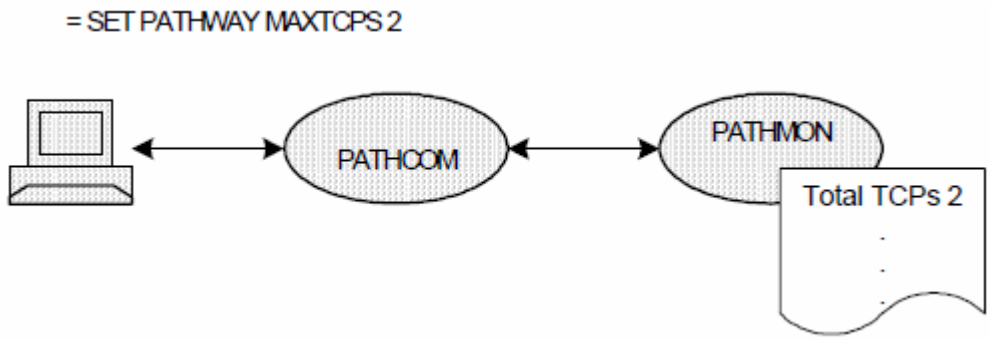

Figure 14: Specifying PATHMON Environment Limits

By specifying limits, you allow the PATHMON process to reserve the disk space that it needs for configuration information. Later, when you actually define and add your PATHMON-controlled objects (described in **Configuring Objects in a PATHMON Environment** on page 50). The PATHMON process monitors these limits and ensures that they are not exceeded.

Specifying Node Independence

Using the NODEINDEPENDENT attribute makes unspecified node names for processes and devices default to the node where the current PATHMON process is running. Using this attribute in your PATHMON configuration facilitates switching your PATHMON 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 resolve to that node, thereby migrating the associated objects.

This command establishes node independence for objects in your PATHMON configuration, unless a node name is explicitly specified for a given object:

```
= SET PATHWAY NODEINDEPENDENT ON
```

To take full advantage of the node-independent feature, you must invoke it by using the SET PATHWAY NODEINDEPENDENT command prior to using the SET PATHMON commands that specify a backup processor and dump file. For more details on how the NODEINDEPENDENT attribute works, see **Configuring Objects in a PATHMON Environment** on page 50.

Note that setting the Pathway NODEINDEPENDENT attribute to ON overrides and disables the node field of the CMDVOL command. For more details, see **CMDVOL Command** on page 145.

Specifying the Owner and Security Attributes

You specify security for your PATHMON environment by setting the OWNER and SECURITY attributes. These attributes determine who is allowed to alter the definitions and states of PATHMON-controlled objects.

For example, these commands indicate that the owner is the super ID and that only the owner can add, modify, or delete PATHMON-controlled objects:

```
= SET PATHWAY OWNER 255,8
= SET PATHWAY SECURITY "O"
```

Configuring the PATHMON Process

Use PATHCOM commands to configure your PATHMON process. For example, you might want to specify a backup processor or request error dumping.

Specifying a Backup Processor

Because the PATHMON process is a crucial control process, it typically runs with a backup process.

To define the backup process's location, use the SET PATHMON command. For example, this command selects processor 1 as the processor for the backup process:

```
= SET PATHMON BACKUPCPU 1
```

You must specify a processor other than the one in which the PATHMON primary process is currently running.

Requesting Error Dumping

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

- Use the SET PATHMON command if you want to request error dumping and you have not yet started your Pathway environment.
- Use the CONTROL PATHMON command if you want to request error dumping but you have already started your Pathway environment. This command is described in more detail in **Changing Backup CPUs and Dump Files** on page 97.

In case of an internal or fatal error, the PATHMON process generates an error dump and writes it to a file. You might specify a name for the dump file. For example, this command directs the PATHMON process to write error information to a file named PMDUMP:

```
= SET PATHMON DUMP ON (FILE PMDUMP)
```

The PATHMON process writes the information in the data stack to PMDUMP.

If you request your Hewlett Packard Enterprise representative's help in analyzing a problem, the representative will likely require a DUMP file. Hewlett Packard Enterprise recommends that DUMP option must be set to ON for production systems.

Specifying the START Command

After you have started the PATHMON process and PATHCOM and configured global parameters, you can start your PATHMON environment.

There are two START command options: COLD and COOL. Which one you choose depends on the task you want to perform:

- Starting a new PATHMON environment (cold start)

  When you start a PATHMON environment for the first time, you must specify the COLD start option. The COLD start option tells the PATHMON process to create the configuration file, PATHCTL, that will store configuration information.

- Restarting a PATHMON environment (cool start)

  When you restart a PATHMON environment, you must specify the COOL start option. The COOL start option tells the PATHMON process to get the configuration information from the existing PATHCTL file. A cool start is usually much faster than a cold start.

Starting a New PATHMON Environment (COLD Start Option)

To start your PATHMON environment for the first time, enter the START PATHWAY command, specifying the COLD start option as follows:

```
= START PATHWAY COLD
```

In response to this command, the PATHMON process performs two steps, as described below and shown in **Figure 15: Starting a New PATHMON Environment** on page 42:

1. Starts the PATHMON backup process if one is specified.
2. Creates the PATHMON configuration file, PATHCTL.

   PATHMON preallocates disk space for the configuration file based on the maximum values that you specified in the `SET PATHWAY` commands prior to cold start.

Depending on the size of your PATHMON environment, the time required to complete a cold start can be lengthy. Consequently, you must use the `COLD start` option only if any of this are true:

- You are starting your PATHMON environment for the first time.
- You have changed the NODEINDEPENDENT, OWNER, or SECURITY attributes.
- You have added SPREBALANCEMODE and/or DUMASK using `SET PATHWAY` command in the Pathway environment.

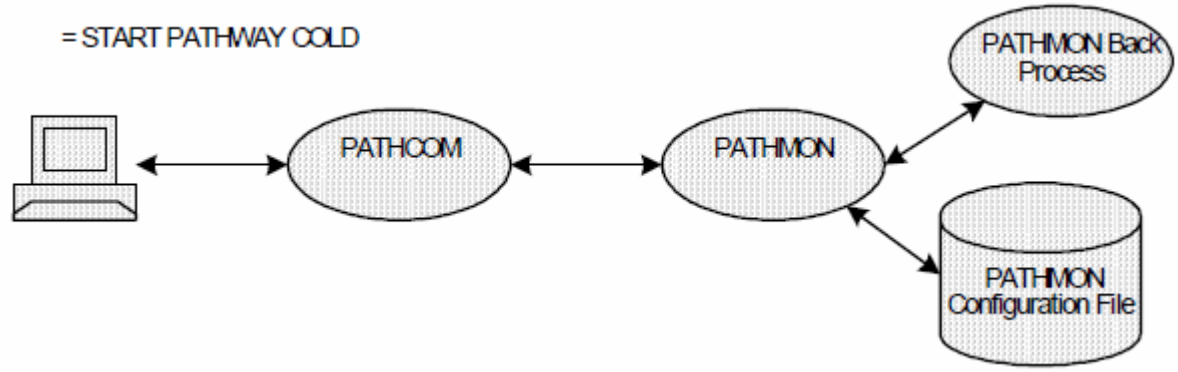

Figure 15: Starting a New PATHMON Environment

Restarting a PATHMON Environment (COOL Start Option)

To restart a PATHMON environment, use the `COOL start` option as follows:

```
= START PATHWAY COOL
```

In response to this command, the PATHMON process:

- Starts the PATHMON backup process if one is specified.
- Uses the existing configuration file as a basis for creating the operating environment. The existing configuration file represents the PATHMON configuration at the time the system stopped (due to shutdown or failure).
- Prompts you, if the PATHMON process name is different from the name stored in the PATHCTL file, that the PATHMON name is changed in the PATHCTL file.

  If you do not want to receive this prompt, you can use the `COOL!` option as follows:

  ```
  = START PATHWAY COOL!
  ```

NOTE: If you have altered the node independence, owner or security attributes, or you have increased or decreased system limits, you must specify a cold start operation to restart the PATHMON environment, as shown in this example:

```
= START PATHWAY COLD!
```

This command directs the PATHMON process to overwrite the existing configuration file with the new information. If you use the `COOL start` option, the PATHMON process uses the old configuration file.

Shutting Down a PATHMON Environment

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

You can choose from these three options: `ORDERLY`, `ABORT`, and `IMMEDIATE`. Each option provides a different level of shutdown, as described in **Effects of SHUTDOWN2 Options** and in the subsections that follow.

In an environment that includes the NonStop TS/MP product, the PATHMON process automatically stops SERVER processes under its control, and then the PATHMON process itself.

In an environment that includes both the NonStop TS/MP product and the Pathway/iTS product, the PATHMON process automatically stops objects in this order: TERM objects, TCP objects, SERVER processes under its control, and finally the PATHMON process itself.

NOTE: This manual provides information specific to commands dealing with objects managed through the NonStop TS/MP product. PATHCOM commands dealing with requestor objects, such as TCP, TERM, and PROGRAM objects, which are controlled through the Pathway/iTS product, are described in *Pathway/iTS System Management Manual*.

Once you specify a `SHUTDOWN2` command:

- 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 **PATHMON Environment Control Commands** on page 159.)
- The PATHMON process logs the start and completion of SHUTDOWN2; it does not log status messages during shutdown.

(The PATHMON process supports two shutdown commands: `SHUTDOWN` and `SHUTDOWN2`. It is recommended you use the newer, `SHUTDOWN2` command for a faster, more reliable shutdown operation. For compatibility reasons, however, the `SHUTDOWN` command is still available. (For more information, see **SHUTDOWN Command (Old)** on page 179.)

NOTE: For shutting down the PATHMON environment in TS/MP 2.6 or later, PDMCOM or PATHCOM can be used. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

Table 2: Effects of SHUTDOWN2 Options

	ORDERLY	ABORT	IMMEDIATE
TERM objects	Stopped	Aborted	Aborted[1]
SERVER processes	Closed	Closed	Stopped[2]
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) and ACS subsystem processes	Notified of shutdown request[3]	Notified of shutdown request[3]	Notified of shutdown request[3]
PATHMON process	Stopped—if shutdown completes without errors	Stopped—if shutdown completes without errors	Stopped—if shutdown completes without errors
Outstanding work	All outstanding work (I/O and dialog activity) is completed	For TCPs and server classes, outstanding work (I/O and dialog 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 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 ACS subsystem processes notifies of an impending PATHMON shutdown, TCP or ACS subsystem processes no longer sends data or requests to the server classes that are under the control of the PATHMON process and are shutting down.

[4] If the transaction is protected by the Transaction Management Facility (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.

Specifying the ORDERLY Option

The `ORDERLY` option allows work in progress to complete before shutting down. This 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.

The ORDERLY option opens and closes unlinked server processes, permitting the performance of any epilog processing.

Specifying the ABORT Option

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

```
= SHUTDOWN2, MODE ABORT
```

The ABORT option allows send statements 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. The ORDERLY option opens and closes unlinked server processes, permitting the performance of any epilog processing.

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. Unlinked server processes are not able to perform any epilog processing.
- The status of any outstanding transactions is unknown.

Escalating the Shutdown Operation

If you choose the ORDERLY or ABORT option, you can escalate the shutdown operation by pressing the Break key and then specifying a faster option.

For example, if you specify ORDERLY, but find that you require a faster shutdown, you can press the Break key and then specify either the ABORT option or, for fastest shutdown, the IMMEDIATE option, as shown in this example:

`= SHUTDOWN2, MODE ORDERLY`	`Shutdown begins. You decide to escalate the shutdown operation, so you press the Break key.`
`=`	`The PATHCOM prompt returns.`
`= SHUTDOWN2, MODE IMMEDIATE`	`You specify the SHUTDOWN2, IMMEDIATE command for the fastest shutdown possible.`
`=`	`Once the shutdown operation completes, the PATHCOM prompt returns.`

NOTE: Pressing the **Break** key does not affect the execution of the shutdown process. It simply returns the PATHCOM prompt to your terminal so that you can enter other commands: for example, `STATUS PATHWAY` or a more forceful `SHUTDOWN2` option. For more information, see **Using PATHCOM During Shutdown** on page 49.

You can also specify a `TIMEOUT` option that helps you escalate SHUTDOWN by returning the PATHCOM prompt within a specified time period. This command specifies a five-minute timeout option:

```
= SHUTDOWN2, MODE ORDERLY, UNTIL TIMEOUT 5 MINS
```

If the shutdown operation has not completed after five minutes, the PATHCOM prompt returns while the shutdown operation continues.

The `TIMEOUT` option is especially useful if you want to escalate the shutdown operation from within a command file or TACL routine.

Monitoring Shutdown Status

You can monitor the status of the shutdown in one of two ways:

- Include the `STATUS AGGREGATE` option in your `SHUTDOWN2` command, as shown in this example:

  ```
  = SHUTDOWN2, MODE ORDERLY, STATUS AGGREGATE, UNTIL TIMEOUT
  5 MINS
  ```

  The `STATUS AGGREGATE` option tells PATHCOM to display shutdown status every 20 seconds until shutdown completes or until the `TIMEOUT` option (if specified) expires.

- Specify the `STATUS PATHWAY` command from any PATHCOM process. (For details on using the PATHCOM that issued the `SHUTDOWN2` command, see **Using PATHCOM During Shutdown** on page 49.) As an example, this command displays status every 30 seconds, up to 5 times:

  ```
  = STATUS PATHWAY, COUNT 5, INTERVAL 30 SECS
  ```

As shown in **SHUTDOWN2 Status Display** on page 46, both `STATUS` commands indicate:

- The Pathway environment is shutting down, or shutdown has failed
- The time shutdown began
- Status of all entities defined in the Pathway environment

SHUTDOWN2 Status Display

PATHWAY -- STATE = SHUTTING DOWN (IN PROGRESS) - STARTED 2:11:39					
	RUNNING				
LINKMONS	0				
PATHCOMS	1				
SPI	0				

Table Continued

	RUNNING	STOPPED	THAWED	FREEZE FROZEN	PENDING
SERVERCLASSES	2	1	1	0	0
	RUNNING	STOPPED	PENDING		
SERVERPROCESSES	8	1	0		
TCPS*	2	4	0		
	RUNNING	STOPPED	PENDING	SUSPENDED	
TERMS*	8	8	0	0	
PATHWAY - STATE = SHUTTING DOWN (IN PROGRESS) - STARTED 2:11:39					
	RUNNING				
LINKMONS	0				
PATHCOMS	1				
SPI	0				
	RUNNING	STOPPED	THAWED	FREEZE FROZEN	PENDING
SERVERCLASSES	0	3	1	0	0
	RUNNING	STOPPED	PENDING		
SERVERPROCESSES	1	8	0		
TCPS*	0	6	0		
	RUNNING	STOPPED	PENDING	SUSPENDED	
TERMS*	0	16	0	0	
* Defined only if you have the Pathway/iTS product					

If Shutdown Cannot Complete

If the PATHMON process encounters an error while shutting down, it terminates the shutdown operation. PATHCOM returns the reason for the failure. In some cases, you might have to examine the log file to determine why shutdown did not complete, fix the problem, and reissue the `SHUTDOWN2` command.

This are examples of situations in which shutdown cannot complete:

- An external object (ASSOCIATIVE), such as a server process, cannot be stopped.
- The network goes down and a locally controlled TCP is running on a remote system in that network.

To see if objects are still running, you must specify a `STATUS PATHWAY` command. If necessary, you can try to stop these objects by specifying a `Guardian STOP` command at your TACL prompt.

Stopping the PATHMON Process

If shutdown cannot complete, you can stop the PATHMON primary and backup processes by issuing this PATHCOM command:

```
= STOP PATHMON
```

This command stops the PATHMON primary and backup processes and writes a termination message to the log file.

NOTE: The `STOP PATHMON` command is valid only if you have already issued a `SHUTDOWN2` command.

When stopping the PATHMON process, you must consider that processes started by the PATHMON process are no longer under control and might still be running. You must find and stop these “orphan” processes. One way to stop orphan processes is to use a TACL routine. **Sample TACL Routine to Stop Orphan Processes (Local System Only)** on page 48 shows a TACL routine that, for a local system, first stops all children of a given process ID and then stops the process ID. For more information on using TACL routines, see the *TACL Reference Manual* and the *TACL Programming Guide*.

Sample TACL Routine to Stop Orphan Processes (Local System Only)

```
?SECTION stop_children ROUTINE
== Stops all children of a given process id & then stops the process id
== USAGE:
== STOP_CHILDREN <process-id>
== EXAMPLE:
== STOP_CHILDREN $CRPM
== this would stop $CRPM and all children of $CRPM (if used by
== appropriate user-id)
== LIMITATION
== This issues explicit stop commands against processes. As such it
== is subject to Guardian security limitations about who can stop
== which processes.
== ENVIRONMENT
== This file is designed to be LOADED into a TACL environment prior
== to invocation.
== AUTHOR: PJL
==
#FRAME
[#PUSH _system _processid _title _entry _result _entry _name
_primary _backup _ancestor _sys _proc _style _outfile]
[#CASE [#ARGUMENT/text _processid/ PROCESSNAME OTHERWISE]
 |1|
  [#IF [#EMPTYV _system] |THEN|
  #SET _system [#FILEINFO/SYSTEM/ [_processid]]
  ]
  [#IF [#EMPTYV _system] |THEN|
    #SET _system [#FILEINFO/SYSTEM/ [#DEFAULTS]]
    [#IF [#EMPTYV _system] |THEN|
      #SET _system [#MYSYSTEM]
```

```
    ]
  ]
  [#IF [#ARGUMENT END]]
  #SET _entry 0

  [#LOOP |DO|
    [#SETMANY _result _entry _name _primary _backup _ancestor,
      [#LOOKUPPROCESS/RESULT,ENTRY,PROCESSID,PRIMARY,BACKUP,
                                    ANCESTOR/ _entry [_system]]
      ]
      [#IF [#MATCH [_processid] [_ancestor]] |THEN|
        STOP [_system].[_name]
      ]
      #SET _entry [#COMPUTE _entry + 1]
    |UNTIL| [#EMPTYV _name]
    ]
    STOP [_system].[_processid]
    |otherwise| ==! do nothing
      [#IF [#ARGUMENT END]] ==CLEANUP UNTIL END OF LINE
]
#UNFRAME
```

Using PATHCOM During Shutdown

When you issue a `SHUTDOWN2` command, the PATHCOM prompt is suspended until shutdown completes, or until you specifically request the prompt to return using one of these methods:

- Press the **Break** key during a `SHUTDOWN2` command. This action returns the PATHCOM prompt to your terminal without affecting execution of the shutdown operation.
- Specify a `TIMEOUT` option with the `SHUTDOWN2` command.

After the PATHCOM prompt returns, you can either initiate communication with another PATHMON process or terminate your interaction with PATHCOM. For example, to open communication with the PATHMON process named $PMY, enter:

```
= OPEN $PMY
```

To end your interaction with PATHCOM, enter:

```
= EXIT
```

After PATHCOM runs this command, the PATHCOM process ceases to exist and control returns to TACL.

Configuring Objects in a PATHMON Environment

Configuration Overview

To configure a PATHMON environment, you specify global limits that define the number of objects in the environment, and you configure the objects that run under the PATHMON process to support your application. (A PATHMON environment consists of objects and processes controlled by the PATHMON process.)

NOTE:

This overview provides a look at how objects and processes defined under the NonStop TS/MP product −the PATHMON process and server processes, for example−are configured in relation to other objects and processes in a PATHMON environment. The same commands are used to configure and manage objects provided by the Pathway/iTS product−TCPs, TERM objects, and PROGRAM objects. See the *Pathway/iTS System Management Manual*.

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 16: PATHMON Environment With a Single Application** on page 50.

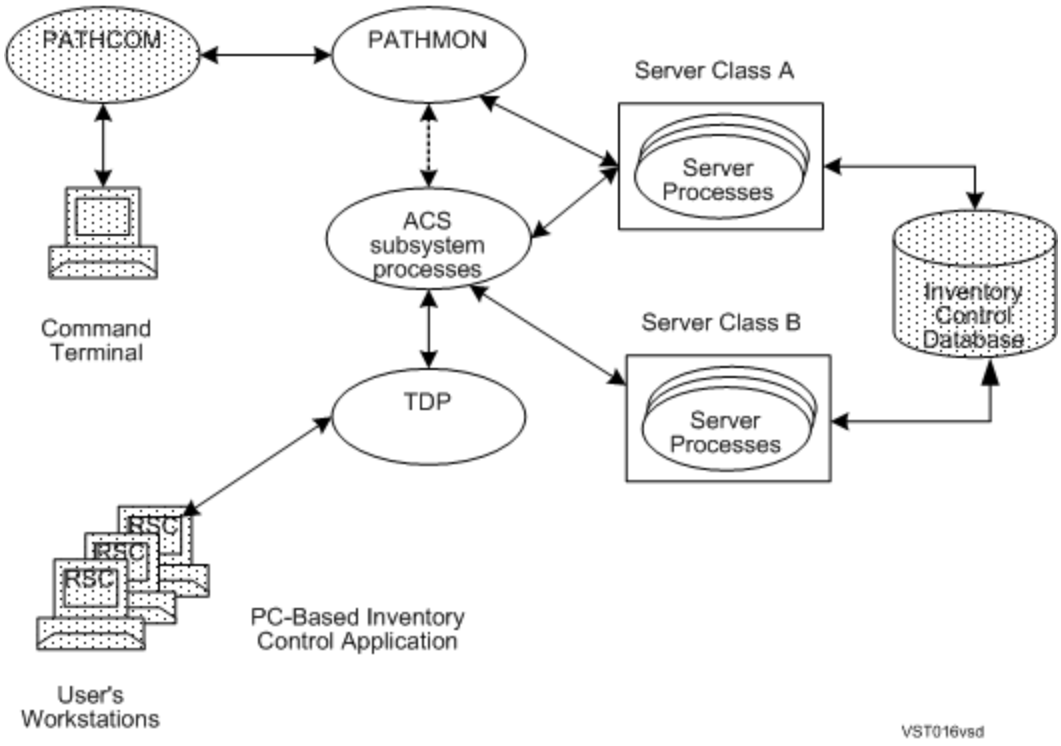

Figure 16: PATHMON Environment With a Single Application

In **Figure 16: PATHMON Environment With a Single Application** on page 50, a PC-based inventory application sends Pathsend requests through the Remote Server Call product (RSC) and Transaction Delivery Process (TDP) to the link manager ACS subsystem processes, which forwards the request to the appropriate server process. The dashed lines indicate control links, while solid lines indicate data links.

A PATHMON environment might also consist of multiple applications such as order processing, accounts payable, accounts receivable, as well as inventory control, as shown in **Figure 17: PATHMON Environment With Multiple Applications** on page 51. In **Figure 17: PATHMON Environment With Multiple Applications** on page 51, each RSC application is configured under its own TDP and all access the same database. Alternatively, each application might have its own database. The dashed line in the figure indicates a control link, while solid lines indicate data links.

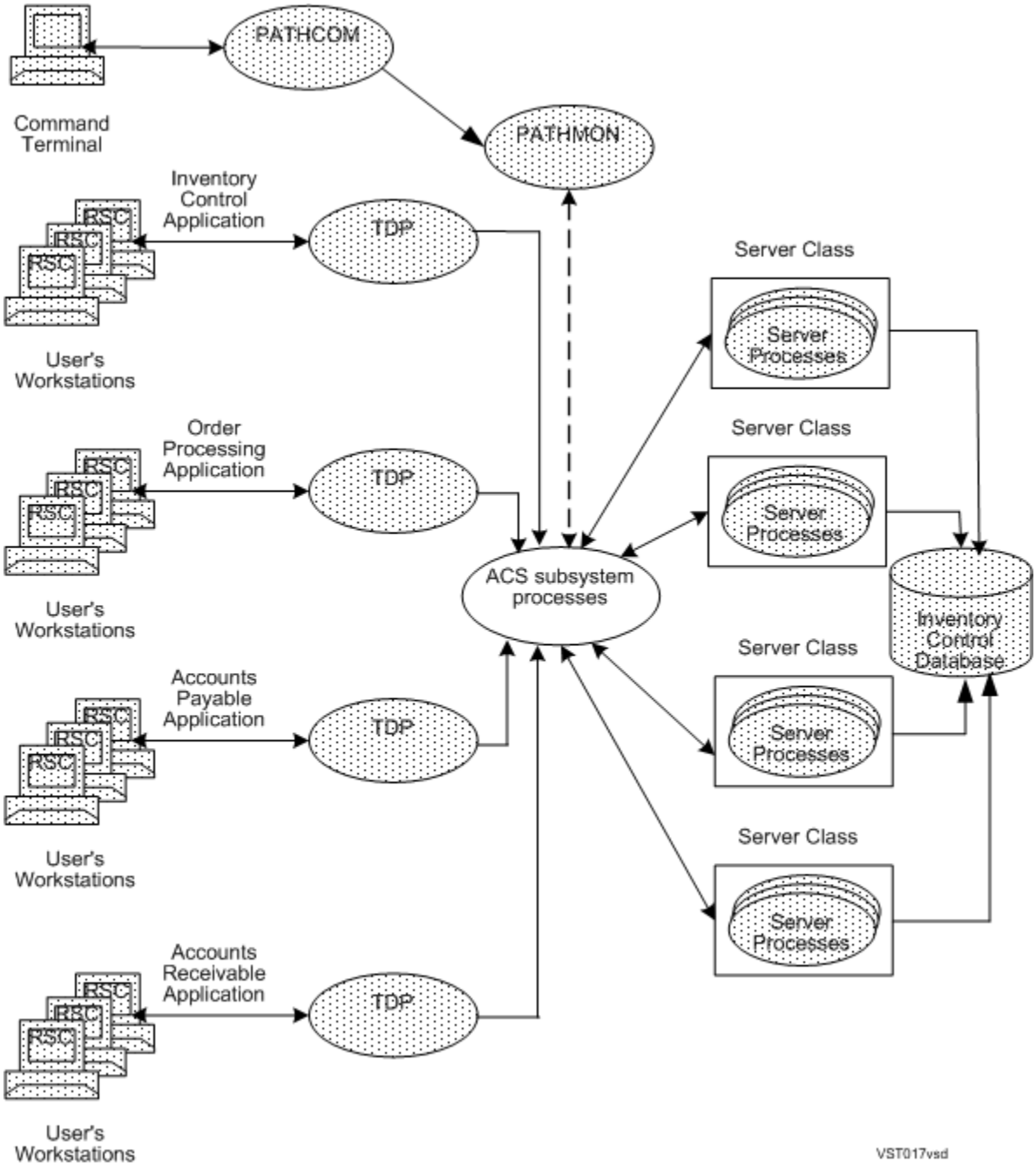

Figure 17: PATHMON Environment With Multiple Applications

In some configurations, a single application spans multiple PATHMON environments that might be distributed over a network. In cases where more than one PATHMON environment is involved, each environment is monitored by the PATHMON process. For more information, see **Distributing a Pathway or PATHMON Environment** on page 18.

Because NonStop systems support both Guardian and OSS operating environments, configurations on these systems can include applications running in both operating environments. For an example of a NonStop TS/MP application configured in two operating environments, see **Figure 2: NonStop TS/MP Application With Guardian and OSS Servers** on page 18. As a system manager for a PATHMON environment, you can configure and manage OSS server objects.

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 NonStop TS/MP application that includes Pathway/iTS objects−for example, 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).

If your application generates a large number of Pathsend requests (like the application shown in **Figure 17: PATHMON Environment With Multiple Applications** on page 51), you must consider distributing the request load by configuring your TDP processes across a number of CPUs. Such an arrangement avoids overwhelming any single ACS subsystem processes; each ACS subsystem process can handle only a certain number of Pathsend requests. (For descriptions of configuration limits, see **Configuration Limits and Defaults** on page 364.)

Configuring a PATHMON Environment

As described in **Starting and Stopping a PATHMON Environment** on page 33 the `SET PATHWAY` command allows you to define the limits for your overall configuration and to specify owner and security attributes for your PATHMON environment. If you anticipate ever needing to migrate your application to another node, you must consider configuring your environment for node independence.

NOTE: For configuring the PATHMON environment in TS/MP 2.6 or later, PDMCOM or PATHCOM can be used. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information, see the *TS/MP 2.7 ACS Reference Manual*.

Specifying Limits

You must use the `SET PATHWAY` command to define this limits before executing a `START PATHWAY` command:

MAXASSIGNS MAXPARAMS MAXSERVERCLASSES MAXSERVERPROCESSES MAXSTARTUPS

The limits that you specify determine the maximum number of SERVER objects you can define for your system during subsequent configuration:

- MAXASSIGNS specifies the number of ASSIGN definitions that you can specify for all server classes.
- MAXPARAMS specifies the maximum number of server classes that can have PARAM messages.
- MAXSERVERCLASSES specifies the maximum number of server-class descriptions that you can add to the PATHMON configuration file.
- MAXSERVERPROCESSES specifies the maximum number of server processes that you can define for all server classes using the `MAXSERVERS` option of the `SET SERVER` COMMAND.
- The total of all MAXSERVERS values for all server classes is limited to the value of MAXSERVERPROCESSES.

You might have to first establish a value for MAXSERVERS for each server class in your application, then adding up all the values to get an appropriate value for MAXSERVERPROCESSES. For information on determining a value for MAXSERVERS, see **Configuring Links for Optimum Performance** on page 67.

- MAXSTARTUPS specifies the maximum number of server classes that can have startup messages.
- 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. Some of the optional attributes are:
  - MAXLINKMONS specifies the maximum number of link manager (LINKMON/ACS subsystem) processes that can communicate with the PATHMON process at the same time.
  - MAXPATHCOMS specifies the maximum number of PATHCOM processes that can run simultaneously within a PATHMON environment.

For a complete description of all the SET PATHWAY attributes, see **SET PATHWAY Command** on page 171.

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

Hewlett Packard Enterprise recommends that you always allow some space for growth. For example, if you are not certain whether you might need an extra server class, leave room for one or two of these objects. 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 the corresponding swap-file space.

For more information about limits and default values for your PATHMON environment, see **Configuration Limits and Defaults** on page 364. Note that identifying a truly optimum configuration for your PATHMON environment can require a significant degree of calculation and tuning that is beyond the scope of this manual. For additional help, contact your Hewlett Packard Enterprise representative for details about performance classes offered and for a description of services available from service provider.

Specifying Node Independence

You can avoid hard-coding node names for many Guardian file names and other variable names in your PATHMON environment by setting the NODEINDEPENDENT attribute of the `SET PATHWAY` command to ON. When NODEINDEPENDENT is ON, unspecified node names for processes and devices default to the node where the PATHMON process is running after cool start.

Specifying node-independent names for processes and devices facilitates switching your PATHMON application to another node, either in the event of failure or as part of a planned migration (for example, in a configuration that uses Nomadic Disk technology). When the PATHMON process is cool started on a different node, the unspecified node names resolve to that node, thereby migrating the associated objects.

This command establishes node independence for objects in your PATHMON configuration, unless a node name is explicitly specified for a given object:

```
= SET PATHWAY NODEINDEPENDENT ON
```

When the NODEINDEPENDENT attribute is ON, any unspecified node names in the Guardian file names defined for processes or devices are automatically set to *, a generic node name designating the node where the PATHMON process is currently running. NonStop TS/MP and Pathway/iTS both support objects that can be configured as node-independent—SERVER objects, for example, as well as TERM and PROGRAM objects and TCPs. For the complete syntax and a list of object attributes containing Guardian file names, see the **SET PATHWAY Command** on page 171.

When to Set Node Independence

If you choose to use the NODEINDEPENDENT attribute, specify it early in your PATHMON configuration session to ensure that it applies to other global parameters, for example, the PATHMON dump file. Once set, the attribute applies to all subsequent `SET` commands. The setting has no effect on previously specified `SET PATHWAY` commands.

Node Independence and the CMDVOL Command

Setting the Pathway NODEINDEPENDENT attribute to ON overrides and disables the node field of the `CMDVOL` command. See **CMDVOL Command** on page 145.

Some Names Are Always Node-Independent

Note that some object names are node-independent regardless of whether you use the NODEINDEPENDENT attribute or not. These include the names of log files in which the PATHMON process records errors and object status changes. If you leave the node portion of a log file name unspecified, the node name always defaults to the node where the PATHMON process is running at any given time.

Migration Considerations

Designating your PATHMON application as node-independent simplifies, but does not eliminate, the tasks associated with migrating an application from one system to another. Migrating device names configured under Pathway/iTS, for example, is not useful unless physical devices with those names are connected to the new node. For considerations when migrating your PATHMON application to a new node, see **Migrating Your Environment to a Different System** on page 109.

Specifying Security

You can specify security for your PATHMON environment by setting the OWNER and SECURITY attributes of the `SET PATHWAY` command.

The OWNER attribute indicates the owner of the PATHMON environment, who can stop the PATHMON process, add programs, delete objects, and so on. The owner can change every attribute of the global configuration.

PATHMON SECURITY values (A, G, O, -, N, C, and U) are the same as Guardian security values.

This example specifies that only the owner—user ID 8,61—can modify the PATHMON environment:

```
SET PATHWAY OWNER 8,61
SET PATHWAY SECURITY "O"
```

The next example specifies that any member of user group ID 8 can modify the PATHMON environment:

```
SET PATHWAY OWNER 8,61
SET PATHWAY SECURITY "G"
```

Before you issue the `START PATHWAY` command, the owner ID is always the process accessor ID of the PATHMON process and the security attribute is O (owner). This setting prevents alteration of the global configuration parameters.

NOTE: After you issue the `START PATHWAY` command, if you do not specify a value for the security attribute, it defaults to O. If the owner of this system is the super ID, the N or A value results in a security risk for your system. You must set the security attribute to O, so that only persons logged on as the super ID can add, delete, or modify PATHMON-controlled objects.

The default security attribute is “O”.

You can also specify security for running programs, as described later in this section.

Configuring PATHMON-Controlled Objects

After specifying global limits and issuing the START command, you configure the SERVER objects that run under the PATHMON process to support your application.

If your environment includes the Pathway/iTS product, you also configure TCP, TERM, and PROGRAM objects. For instruction on configuring these objects, see the *Pathway/iTS System Management Manual*.

You create and control objects by defining and changing their attributes. PATHCOM maintains a list of attributes that describe configuration information for each object: how it relates to other objects and how it must be managed by the PATHMON process. The attributes for a server, for instance, specify the server name, the server type (Guardian or OSS) the CPUs on which the server runs, the server's execution priority, the name of the program the server runs, and other characteristics.

NOTE: For configuring PATHMON-controlled objects in TS/MP 2.6 or later PATHMON environment, PDMCOM or PATHCOM can be used. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information, see the *TS/MP 2.7 ACS Reference Manual*.

Using the SET and ADD Commands

Whether you create objects for a new environment or add objects to a restarted environment, the technique is the same, as follows:

1. Define the attributes for the object, using a SET command parameter for each attribute.

   As you define these attributes, PATHCOM checks them for syntax errors. PATHCOM saves the attributes in memory, pending your request to add the object. In many cases, you can choose to accept default values supplied by PATHCOM.

   When creating objects, be sure to set proper defaults so that objects are created in the appropriate location. When creating Guardian server classes, use the CMDVOL command to set the default node, volume, and subvolume for the expansion of any file names. When creating OSS servers, use the CMDCWD command to record an OSS directory to be used to resolve relative file names specified for OSS server attributes. For more details on these commands, see **CMDVOL Command** on page 145 and **CMDCWD Command** on page 144.

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

   - You set the NODEINDEPENDENT attribute of the 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 more information on the NODEINDEPENDENT attribute, see the description of the SET PATHWAY command in **PATHMON Environment Control Commands** on page 159.

2. Add the object with the ADD command.

   The PATHMON process writes the object's name and definition to the PATHMON configuration file, inserting this information in the space allocated for it when the file was created. The object definition

recorded in the configuration file includes attribute values assigned by default as well as those that you specified in your SET commands.

The object name you assign in the ADD command is used in later PATHCOM commands that refer to the object. For more information about names, see **Choosing Names for PATHMON-Controlled Objects** on page 62.

NOTE: You cannot configure remote OSS servers in your PATHMON configuration. Although PATHCOM allows you to ADD an OSS server configured on a remote node, PATHMON cannot START a remote OSS server.

To illustrate the relationship between the SET and ADD functions, suppose that you want to define and add a SERVER identified by the name SERVER-X. You first define the attributes by using a series of SET SERVER commands or by accepting default attribute values from PATHCOM:

```
= SET SERVER PROCESSTYPE OSS
= SET SERVER CREATEDELAY 1 MINS
= SET SERVER MAXSERVERS 2
= SET SERVER CPUS (3:2, 0:6)
= SET SERVER PROGRAM \PARIS.$MARKT.UPDATE
= SET SERVER SECURITY "N"
```

Before adding the SERVER, you can check the values by using the SHOW command:

```
= SHOW SERVER
```

Then, you name and add the SERVER, using an ADD SERVER command:

```
= ADD SERVER SERVER-X
```

In response, the PATHMON process writes the name and definition for this SERVER into the PATHMON configuration file, as shown in **Figure 18: Defining and Adding a PATHMON-Controlled Object** on page 57.

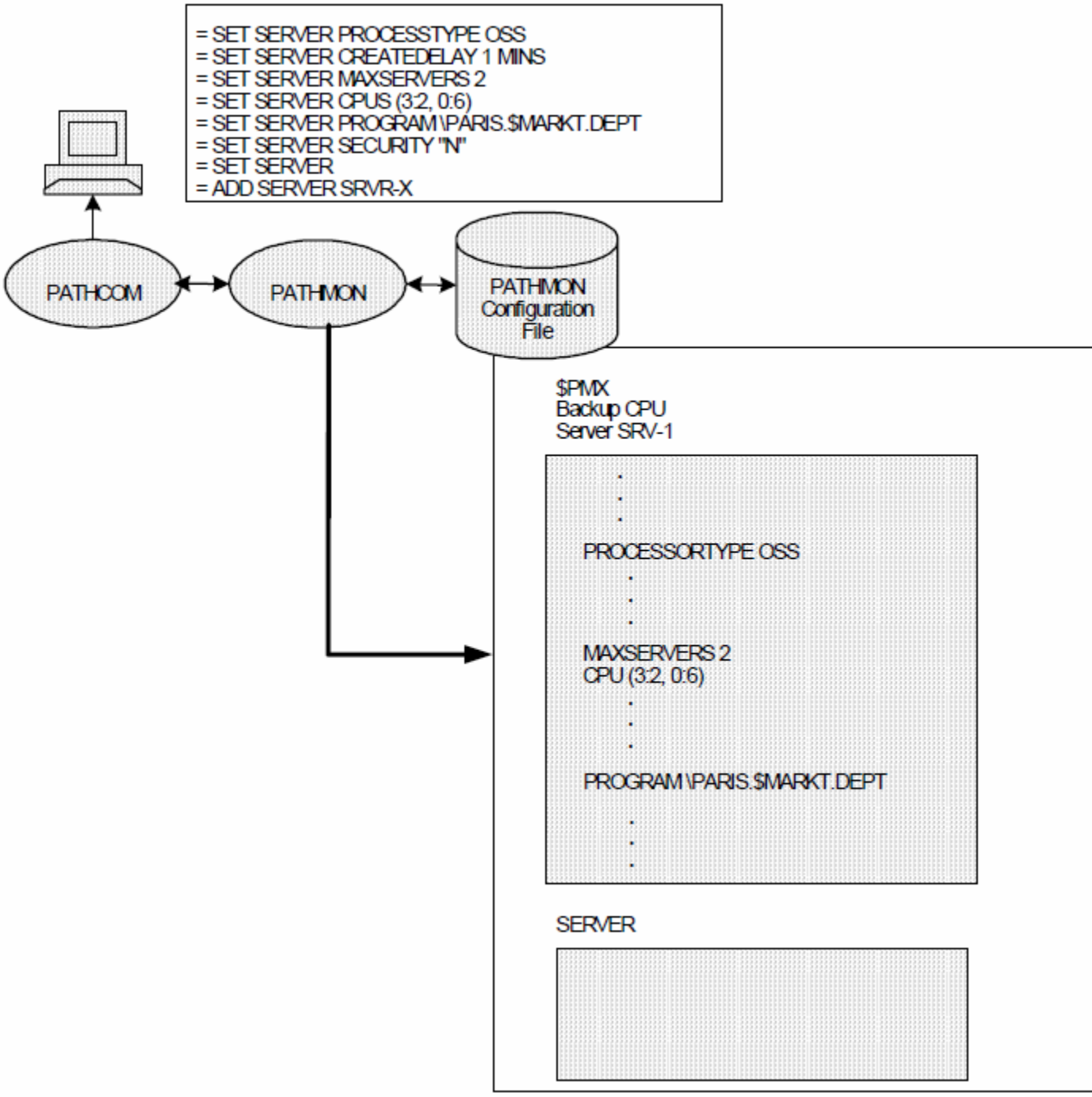

Figure 18: Defining and Adding a PATHMON-Controlled Object

Understanding the Working Set of Attribute Values

PATHCOM maintains a working set of attribute values for objects you define in your PATHMON environment. These objects include SERVER objects and −if your environment includes the Pathway/iTS product−TCP, TERM, and PROGRAM objects. For information about managing Pathway/iTS objects, see the *Pathway/iTS System Management Manual*.

The working set of attribute values is initialized when you start PATHCOM. In this initial stage, each attribute entry in the working set specifies either of these:

- The attribute and the default value, assigned by PATHCOM. If you do not specify another value for this attribute with the SET or ADD command, this default value is assigned when you add any object of this type.
- The attribute alone, with no value specified. You must assign a value for this attribute with the SET or ADD command.

By using values from the PATHCOM working set whenever possible, you can greatly simplify object definition and creation. For example, this command creates a SERVER named FRED, using the current values from the working set:

```
= ADD SERVER FRED
```

NOTE: Do not use PATHCOM-reserved words to identify PATHMON-controlled objects. For a list of reserved words, see **PATHCOM Reserved Words** on page 360.

These paragraphs describe how to display, modify, override, and reinitialize working set values.

Displaying Working Set Values

Before you actually begin configuring objects in your environment, you can display the initial working set for each object type by entering the SHOW command. For instance, to list the attributes in the working set for SERVER objects, you enter:

```
= SHOW SERVER
```

In response to this command, PATHCOM displays:

```
SERVER

PROCESSTYPE GUARDIAN
AUTORESTART 0
CREATEDELAY 1 MINS
DEBUG OFF
DELETEDELAY 10 MINS
HOMETERM $term-name
HIGHPIN OFF
LINKDEPTH 1
MAXSERVERS 1
NUMSTATIC 0
OWNER owner
PRI priority
PROGRAM ?
SECURITY "N"
TMF OFF
VOLUME \node.$volume.subvolume
```

In this example, the working set is in the initial state. For attributes with default values assigned by PATHCOM, the default value is displayed; for example, the default value for the AUTORESTART attribute is 0. Attributes to which you must assign values are flagged with a question mark.

Establishing and Modifying Working Set Values

As you configure the objects in your PATHMON environment, you can establish or modify the attribute values in the PATHCOM working set using the SET command. For example, you can establish a value for the PROCCESSTYPE attribute for SERVER objects by entering:

```
= SET SERVER PROCESSTYPE OSS
```

As another example, you can specify the maximum number of static server processes allowed within a server class by entering:

```
= SET SERVER NUMSTATIC 2
```

Each time you enter the ADD command to add an object, the PATHMON process assigns the working-set values last established for that object type (unless you override these values with new ones in the ADD command). This feature allows you to use the PATHCOM working set as a template for defining and adding multiple objects of the same type with identical or similar attribute values.

For example, these commands add four SERVER objects to your PATHMON environment.

```
= SET SERVER PROCESSTYPE GUARDIAN
= SET SERVER LINKDEPTH 1
= ADD SERVER FRED, PROGRAM \*.$MKT.SALES.HIST
= ADD SERVER JOAN, PROGRAM \*.$MKT.SALES.PROJ
= SET SERVER LINKDEPTH 2
= ADD SERVER DON, PROGRAM \*.$MKT.SALES.ANAL
= ADD SERVER MARY,PROGRAM \*.$MKT.SALES.ADMIN
```

All these SERVER objects have the same PROCESSTYPE attribute (namely, Guardian). FRED and JOAN have a LINKDEPTH of 1, while DON and MARY have a LINKDEPTH of 2. Each SERVER object has a unique, node-independent, PROGRAM attribute. **Figure 19: Addition of SERVER Attributes to the PATHMON Configuration File** on page 60 shows the attribute values for the SERVER object named MARY being added to the PATHMON configuration file.

The PATHMON process does not check that you have assigned all the required attribute values for an object until you enter the ADD command.

Remember that these values are known only to PATHCOM and are maintained only during PATHCOM execution; when you exit from PATHCOM, they revert to their original settings. For example, for SERVER objects, the value for the PROGRAM attribute ceases to exist when you exit from PATHCOM.

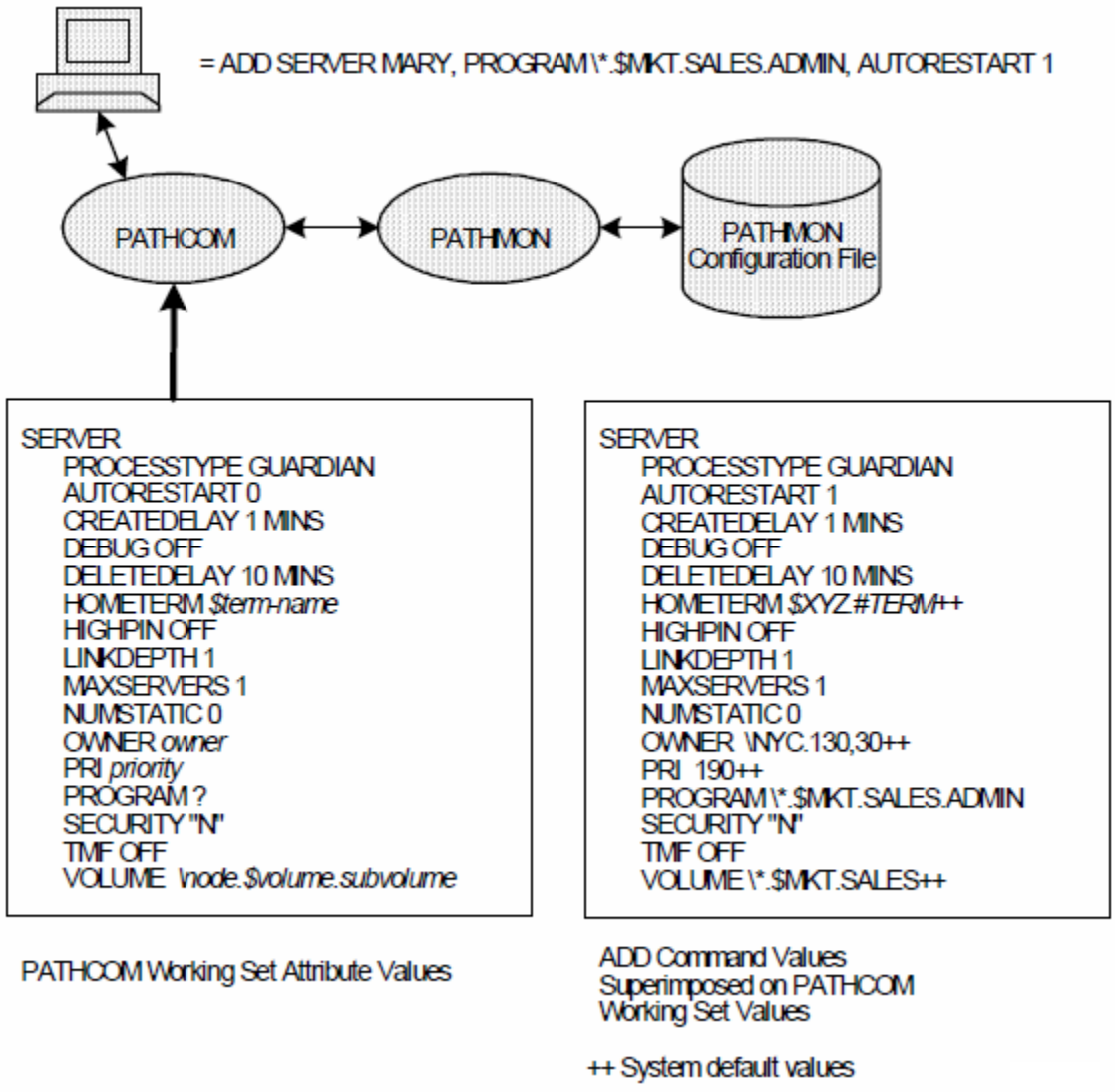

Figure 19: Addition of SERVER Attributes to the PATHMON Configuration File

Overriding Working-Set Values

When you add an object, you can temporarily override the current attribute values in the PATHCOM working set by specifying other values in the `ADD` command. For example, this command creates a SERVER named JOHN, using the working-set values for all SERVER attributes except the HIGHPIN attribute (which will have the value ON):

```
= ADD SERVER JOHN, HIGHPIN ON
```

Notice that this command does not change the values currently recorded in the working set—it only overrides them for this particular instance of the `ADD` command.

Reinitializing Working-Set Values

You can reset any or all PATHCOM working-set values to their initial state (the values that existed when PATHCOM was started) by using the `RESET` command. For example, to reset the PROCESSTYPE, AUTORESTART, and NUMSTATIC values for SERVER objects to their initial settings, you enter:

```
= RESET SERVER PROCESSTYPE, AUTORESTART, NUMSTATIC
```

To reset all SERVER attributes to their initial values, you enter:

```
= RESET SERVER
```

If an attribute for an object has no standard initial default value, PATHCOM resets that attribute to indicate that the value is not yet specified (you see the attribute flagged with a question mark when you issue the SHOW command).

Displaying Object Attributes

After an object is added to the system and the attributes are recorded in the PATHMON configuration file, you can examine these attributes by using the INFO command. For instance, to display the attribute values for the SERVER named ANNA, you would enter:

```
= INFO SERVER ANNA
```

The resulting display appears in this format:

```
SERVER ANNA

PROCESSTYPE GUARDIAN
AUTORESTART 0
CPUS (2:1,3:2,0:1)
CREATEDELAY 1 MINS
DEBUG OFF
DEFINE =EMP, CLASS MAP, FILE \SYS.$D.APPL.EMP
DELETEDELAY 10 MINS
HIGHPIN OFF
HOMETERM \*.$TERM
LINKDEPTH 1
MAXSERVERS 5
NUMSTATIC 2
OWNER \TS.8,8
PRI 134
PROGRAM \*.$BANK1.TEST.CHECK
SECURITY "N"
TMF ON
VOLUME \*.$BANK1.TESTING
```

In the above example, the generic node name * specified for the HOMETERM, PROGRAM, and VOLUME attribute values indicates that node name for these attributes will default to the node where the PATHMON process is currently running.

In **Capturing a Configuration** on page 98 you will see how the INFO command can be used to create OBEYFORM files that contain all the definitions for your complete PATHMON environment. These files can be used as a basis for starting operations.

Using Existing Object Attributes for New Objects

You can use object definitions recorded in the PATHMON configuration file as the basis for defining values in the PATHCOM working set. For example, suppose that you want to add a new SERVER object named JACK, using the same attributes that applied to the SERVER named ANNA (except for the TMF attribute). You could do this by using the SET command with the LIKE option, as follows:

```
= SET SERVER LIKE ANNA
= ADD SERVER JACK, TMF ON
```

The LIKE option requests PATHCOM to set the working-set values exactly the same as those for the attributes of the object whose name follows LIKE. This feature is useful for defining and adding multiple objects of the same kind with similar definitions. For example, when specifying multiple SERVER objects, you need enter only those attributes that are different from those of a previously added SERVER. Alternatively, you can define a SERVER that is exactly like a previously added SERVER.

Choosing Names for PATHMON-Controlled Objects

When adding multiple objects of any one kind, you must name them in a logical way that makes them easy to identify and manage later. For example, you might name SERVER processes according to the functions they perform: ORDERS for a SERVER that processes orders or INV for one that updates inventory control records.

For large PATHMON environments with numerous objects, the names you assign are particularly important for identifying and managing objects. The names you choose can help you identify and relate objects when displaying information about your PATHMON environment.

When you issue a command that uses an asterisk (*) to operate on an entire class of objects−for example, `STOP SERVER *`−the objects are processed in alphabetic order.

Names for SERVER objects can contain from 1 to 15 alphanumeric or hyphen characters. Names must start with a letter; must be unique within the PATHMON environment; and cannot be a PATHMON reserved word.

Configuring Server Classes

The SERVER object represents a server class. A server class is a group of processes, all of which run the same object program. A server process is an executing server program.

If your system supports both the Guardian and the OSS operating environments, you can configure both Guardian server classes and OSS server classes. (To support both operating environments, your system must be a RISC-based, NonStop server and it must have the OSS software running.) Guardian and OSS servers operate the same way and share many attributes. There are some differences in how they are configured, however. These differences are outlined in this subsection.

You use the `SET SERVER` and `ADD SERVER` commands to define the attributes for a server class, name the server class, and add it to the system.

If your system has a NonStop TUXEDO system running in the OSS operating environment, you might also have installed the Pathway translation server for NonStop TUXEDO. This special purpose OSS server allows SCREEN COBOL requestors to access NonStop TUXEDO application services indirectly, through the TCP. (The server can also be used by Pathsend requestors.) The translation server is described in the *Pathway Translation Server for the NonStop TUXEDO System Manual*.

NOTE: PDMCOM or PATHCOM can be used for configuring server classes in TS/MP 2.6 or later PATHMON environment. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information, see the *TS/MP 2.7 ACS Reference Manual*.

Configuring Static and Dynamic Server Processes

There are two types of server processes: static and dynamic. A static server is a server process that the PATHMON process creates when the `START SERVER` command is issued. A dynamic server is a server process that the PATHMON process creates after a link manager has waited for a specific time period for a static server to become available. The time period is determined by the CREATEDELAY attribute for the server class. This type of server is not dependent on the `START SERVER` command.

A dynamic server process runs as long as a link manager ACS subsystem processes, communicates with it. Unused links to dynamic servers are returned to the PATHMON process by the link manager. The PATHMON process deletes unused links based on the time limit defined by the DELETEDELAY attribute for the server class. If high system performance is required, it is recommended that you not use dynamic server processes because they incur a startup penalty as part of the transaction path.

You determine the total possible number of static and dynamic server processes in your system when you specify the `SET SERVER` command, using the NUMSTATIC and MAXSERVERS attributes.

Defining Attributes for Guardian and OSS Servers

For any server class, the PATHMON process requires you to specify the PROGRAM attribute. This attribute indicates the name of the file that contains the object program that the server processes run. (An important difference between Guardian and OSS servers is that whereas you can START Guardian servers on remote nodes, you cannot START remote OSS servers. Thus, a Guardian server PROGRAM attribute can include a remote node name, but an OSS server PROGRAM attribute cannot.)

In addition, the PATHMON process allows you to specify various optional attributes for server classes or to accept default values for them. Most of these attributes are common to both Guardian and OSS servers. Among the optional attributes that apply to both Guardian and OSS servers are:

- PROCESSTYPE specifies whether the server process is a Guardian or an OSS server. The default is Guardian. When you use the `INFO SERVER` command, this attribute is listed first so you can quickly identify a server as a Guardian or an OSS server. All other attributes are listed in alphabetic order.
- AUTORESTART specifies the number of times the PATHMON process attempts to restart a server process within a fixed, 10-minute interval after an abnormal termination, such as a call to the Guardian ABEND procedure. The default is 0. In such cases, the PATHMON process does not attempt to restart a server process after an abnormal termination.

  NOTE: The PATHMON process might restart a static server process any number of times if no other server process is available to serve a current link request irrespective of AUTORESTART attribute.

  If the server process stops under normal circumstances, for example, in response to a `STOP` command, the PATHMON process does not attempt to restart it.

- The CPUS attribute specifies the CPUs on which server processes of this server class will run.

  You can specify either a list of paired (primary and backup) CPUs or a list of single CPUS.

  If you specify single CPUs and a processor is down, the PATHMON process chooses the next processor in the list.

  When you specify a processor pair for a Guardian server, the PATHMON process creates an extra parameter called *BACKUPCPU* to hold the number of the processor specified as a backup. This parameter is passed to the server as part of the PARAM message. As it creates the server processes, the PATHMON process uses the primary CPUs in the order specified. If a processor is down, the PATHMON process uses the backup processor as an alternate processor and passes the primary processor number as the backup processor parameter.

- CREATEDELAY specifies the maximum amount of time, ACS subsystem processes or TCP process must wait to use an established link to a server class before the PATHMON process starts another server process.
- DELETEDELAY specifies the maximum amount of time a link between ACS subsystem processes, or TCP process and a dynamic server process can remain idle before the ACS subsystem processes or TCP process automatically returns the link to the dynamic server.

  The CREATEDELAY and DELETEDELAY attributes have the potential to dramatically impact system performance. See **Configuring Links for Optimum Performance** on page 67.

- DEFINE assigns a DEFINE definition as part of a server class definition. You can use DEFINEs to specify the default volume and subvolume and to specify files used by the server.
- MAXSERVERS specifies the maximum number of server processes in this server class that can be run simultaneously.

- MAXLINKS specifies the maximum number of links to a server process from all TCPs and ACS subsystem processes.
- NUMSTATIC specifies the maximum number of static server processes allowed within this server class. Static server processes are those that start in response to the `START SERVER` command.

For a complete description of all the SET SERVER attributes, see **SERVER Commands** on page 194.

Attributes That Apply Only to Guardian Servers

These attributes are valid only for Guardian servers:

- ASSIGN assigns a logical file name to a physical file used by your server program and specifies the attributes of that file.

  You can include multiple ASSIGN specifications for a server class to cover as many files as necessary. The ASSIGN specification sets up a correspondence between the name used for a file by your server program and the name by which the TACL command recognizes this file. If your server program is written in COBOL, you use this attribute to override a program file assignment made in the SELECT clause. Among the file attributes that you can specify with ASSIGN are disk extent size, access exclusion mode, file code, and record and block size.

- IN specifies the name of the input file passed to the server in the startup message. It can be a DEFINE name. If you omit this attribute, spaces are passed to the server.
- OUT specifies the name of the OUT file passed to the server in the startup message. It can be a DEFINE name. If you omit this attribute, spaces are passed to the server.
- PARAM assigns a string value to a parameter name, which the PATHMON process then sends in a PARAM message to each server process as it starts. You can include multiple PARAM specifications for a server class, just as you can include multiple ASSIGN specifications. (If you specified a backup processor for the processor attribute, the PATHMON process automatically adds a PARAM for the backup processor to the PARAM message.)
- STARTUP specifies a character string sent to the server in the startup message.
- VOLUME specifies the volume and subvolume names to be passed to the server in the startup message. If you omit this attribute, the defaults are those set with the `CMDVOL` command.

These commands defines a Guardian SERVER object and then adds it under the server class name ORDER-SRV:

```
= RESET SERVER
= SET SERVER PROCESSTYPE GUARDIAN
= SET SERVER ASSIGN ORDERS, $DATA.REG1.ORDFILE
= SET SERVER ASSIGN PARTS, $DATA.REG2.PARTFILE
= SET SERVER CPUS (2:4,3:5,0:6)
= SET SERVER MAXSERVERS 5
= SET SERVER NUMSTATIC 3
= SET SERVER PARAM SWITCH-1 "ON"
= SET SERVER PROGRAM \SYS.$DATA.SRVRS.ORDSRV
= ADD SERVER ORDER-SRV
```

The `SET` commands in the preceding example establish these characteristics for the SERVER:

- This is a Guardian server. (PROCESSTYPE attribute.)
- The files that your server program recognizes by the names ORDERS and PARTS are the files $DATA.REG1.ORDFILE and $DATA.REG2.PARTFILE, respectively. (ASSIGN attribute.)

- The PATHMON process starts the first server process in the server class in processor 2, the second server process in processor 3, and the third in processor 0.

  If processor 2 is down, the PATHMON process instead starts the first server process in processor 4 as the backup processor for the first process. Similarly, CPUs 5 and 6 are the backup CPUs for the second and third server processes, respectively. If the PATHMON process starts more than three server processes, it reuses the CPUs in the order specified: for example, a fourth server process would start in processor 2 if that processor were available. (CPUS attribute.)

- The total number of all server processes in the server class that the PATHMON process can run simultaneously is five. (MAXSERVERS attribute.)
- The total number of static server processes allowed is three. (NUMSTATIC attribute.)
- When the PATHMON process starts a server process, it transmits this PARAM message:

  `SWITCH-1 "ON"`

  (PARAM attribute.)

- The object code for the server class is stored in the file `\SYS.$DATA.SRVRS.ORDSRV`. (PROGRAM attribute.)

Attributes That Apply Only to OSS Servers

These attributes are valid only for OSS servers:

- ARGLIST specifies a set of process startup parameters.
- CWD specifies the current working directory to be used to resolve relative OSS filenames specified for server attributes such as PROGRAM, STDIN, STDERR, and STDOUT. The current working directory pathname is combined with a relative pathname to create an absolute OSS filename. (The OSS concept of an absolute filename is similar to the Guardian concept of a fully qualified file name that includes system, volume, and subvolume.)

  An OSS absolute filename can be up to 1024 characters long.

  If you have issued the `CMDCWD` command to establish a working directory, you do not need to set the CWD attribute. However, if you have issued the `CMDCWD` command, you can set the CWD attribute for a given server class to change the working directory just for that server class.

- ENV specifies environmental variables for the server processes in this server class.
- STDERR specifies an OSS filename or device name to which errors are sent.
- STDIN specifies an OSS filename or device name from which this server process accepts input.
- STDOUT specifies an OSS filename or device name to which this server process sends output.

These commands defines an example of an OSS SERVER object and then adds it under the server class name ORDER-SRV1:

```
= RESET SERVER
= SET SERVER PROCESSTYPE OSS
= SET SERVER ARGLIST -1, process/log, -t
= SET SERVER CPUS (2,3,0)
= SET SERVER CWD /centrl/inven/orders
= SET SERVER DEFINE =PARTS, CLASS MAP, FILE $DATA.REG2.PRTFIL
= SET SERVER DEFINE =ORDERS,CLASS MAP, FILE $DATA.REG2.ORDFIL
= SET SERVER ENV DEBUGLOGFORMAT=TRUE
= SET SERVER ENV HANGAROUND=TRUE
```

```
= SET SERVER MAXSERVERS 4
= SET SERVER NUMSTATIC 3
= SET SERVER PROGRAM app/bin/ordsrv1
= SET SERVER STDERR process/error
= SET SERVER STDIN process/orders/new
= SET SERVER STDOUT process/orders/out
= ADD SERVER ORDER-SRV1
```

- This is an OSS process. (PROCESSTYPE attribute).
- As shown by the -1 and process/log values in the arglist, the process logs requests and responses in an OSS file with the absolute filename `/centrl/inven/orders/process/log`. (The server code must get the OSS current working directory information from the CWD attribute and add it to the filename “process/log” to produce the full, absolute pathname.)

  The -t value indicates that a timestamp is associated with each request and response. (ARGLIST attribute.)

  NOTE: The server program name is always passed as the first argument in the arglist, regardless of what you specify in the arglist.

- The PATHMON process starts the first server process in the server class in processor 2, the second server process in processor 3, and the third in processor 0. If the PATHMON process starts more than three server processes, it reuses the CPUs in the order specified; a fourth server process, for instance, would start in processor 2 if that processor were available.

  If processor 2 is down, the PATHMON process instead starts the first server process in processor 3. If processor 3 is down, the PATHMON process starts the first server in processor 0. (CPUS attribute)

- The OSS current working directory for this server process is /centrl/inven/orders. Relative filenames like those specified for the PROGRAM and STDERR, STDIN, and STDOUT attributes are resolved against this directory name to create an absolute pathname. (CWD attribute.)
- PARTS is defined as the logical filename for the physical file located at $DATA.REG2.PRTFIL. ORDERS is defined as the logical filename for the physical file located at $DATA.REG2.ORDFIL. (DEFINE attribute).
- A variable called DEBUGLOGFORMAT is defined. The value is TRUE. (ENV attribute.)

  A variable called HANGAROUND is defined. The value is TRUE. (ENV attribute.)

- The total number of all server processes in the server class that the PATHMON process can run simultaneously is four. (MAXSERVERS attribute.)
- The total number of static server processes allowed is three. (NUMSTATIC attribute.)
- The object code for the server class is stored in the OSS file /centrl/inven/orders/app/bin/ordsrv1. (PROGRAM attribute.)
- Errors are sent to a file with the absolute pathname /centrl/inven/orders/process/error. (STDERR attribute.)
- Input is received from a file with the absolute pathname /centrl/inven/orders/process/orders/new. (STDIN attribute.)
- Output is sent to a file with the absolute pathname /centrl/inven/orders/process/orders/out. (STDOUT attribute.)

Additional Considerations

Following are some configuration considerations that can affect system performance.

Configuring OSS Servers for Effective Space Allocation

When you start the PATHMON environment, the PATHMON process allocates enough memory and disk space to support most configurations. If your environment uses all allocated memory and disk space, the PATHMON process attempts to get more.

You will not experience any increase in your memory and disk space requirements if you configure an environment supporting only Guardian servers; an environment supporting Guardian servers and a few large OSS servers; or an environment supporting Guardian servers and many small OSS servers. On the other hand, if you configure a large number of OSS servers with close to maximum-length attributes, you will see memory and the PATHCTL file increase accordingly.

In rare instances, the PATHMON process might use up its allocated space and request more space from the Guardian environment; or all pre-allocated extents for the PATHCTL file might become full. If the PATHMON process uses all the allocated space, it cannot perform any system functions while it attempts to allocate more space. When the PATHMON process completes the attempt, a message is logged explaining that the PATHMON process was trying to resize the segment.

If all preallocated extents for the PATHCTL file become full, the PATHMON process attempts to allocate another extent. If additional space is not available and a command is issued, the command fails and you get an allocation error.

Configuring for the Use of High PINs

If you have configured your system to 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 server classes as static, thus reducing the overhead associated with process startup.

Configuring Server Processes in Multiple CPUs

Running server processes in different CPUs often increases application throughput, because concurrent copies of the server code can run in parallel.

In assigning CPUs for your server processes, try to balance these processes evenly across the CPUs to avoid overloading any particular processor. Using Hewlett Packard Enterprise performance tools can simplify this process. For a discussion of these tools, see **Other System Management Tools** on page 31.

Setting Process Priority

You can specify the priority at which the PATHMON process, server classes, and TCPs run. To specify the priority at which the PATHMON process runs, include the `PRI run` option in your `TACL` command to start the PATHMON process. In general, the PATHMON process must run at a higher priority than the rest of the PATHMON environment. To specify the priority at which a server class or TCP runs, include the `PRI` option with the `SET TCP` or `SET SERVER` command, respectively.

NOTE:

Your environment includes TCPs only if you have the Pathway/iTS product.

For server classes and TCPs, priorities must facilitate the completion of work over the arrival of work. Usually, this means that server classes must have a higher priority than TCP objects.

Configuring Links for Optimum Performance

Arriving at an optimum system configuration can be challenging because so many variables are involved. The values defined for these variables play an important role in determining whether an application provides optimum, average, or unacceptable performance. This subsection outlines an approach to

configuring key variables in your application environment. The approach, which emphasizes the importance of link management, is presented in a series of steps that you to calculate appropriate values for key variables affecting a given server class. You must repeat most of the steps for each server class in your application, then add the totals to determine global limits for such attributes as MAXSERVERPROCESSES.

Note that although the calculations and formulas that follow are intended to assist you in arriving at an optimum configuration, they are necessarily generic. Only you know the specific characteristics of your system. As a best practice, when utilizing these formulas is that you will pay a small penalty—in unused resources—if you overconfigure your system. On the other hand, you will pay a large penalty—in performance and operations problems and obscure system errors— if you underconfigure your system.

Links and Link Attributes

A link is a connection between a link manager, that is the ACS subsystem processes, and a specific server process. The link is used to send a request to, and receive a reply from, a server.

ACS subsystem processes share a link among the multiple Pathsend processes executing in the processor of the ACS subsystem processes. Only one Pathsend process at a time can use the link (for example, by calling the SERVERCLASS_SEND_ procedure).

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

- If the link manager can get a static link, it asks for a link immediately.
- If the process cannot get a static link, and no links become available in the period of time specified in the *CREATEDELAY* parameter, then the link manager asks for a link to a dynamic server process. Dynamic server processes are temporary processes that the PATHMON process starts as needed and that run only as long as the TCPs or ACS subsystem processes require them.

The PATHMON process will use all existing server processes (both static and dynamic) before starting another server process. First, all links for the static server processes are allocated. Then for each dynamic server process, all links available to the server process (indicated by MAXLINKS) are granted prior to starting another server process. This can lead to contention for the dynamic server process.

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).

You can change the number of links by changing the server class attribute MAXLINKS or, preferably, you can add static servers. It is recommended that you configure for enough static servers so that your application will need to employ few, if any, dynamic servers.

At startup time, to ensure that all static servers are started and, consequently, links are evenly spread, issue a `START SERVER *` command prior to starting any Pathsend processes or, if you use Pathway/iTS, any TERM objects.

To specify information about links, use these attributes of the `SET SERVER` command:

- **MAXSERVERS**
- **MAXLINKS**
- **LINKDEPTH**
- **NUMSTATIC**
- **CREATEDELAY and DELETEDELAY**

The PATHMON process creates server processes to create links. Consequently, the value you specify for the `SET PATHWAY MAXSERVERPROCESSES` command also affects your link configuration. For more information, see **Specifying Limits** on page 52.

MAXSERVERS

The MAXSERVERS attribute specifies the maximum number of server processes in a server class that can run at the same time. To determine maximum number of links available in a server class, use the formula MAXSERVERS x MAXLINKS.

MAXLINKS

The MAXLINKS attribute specifies the maximum number of links to an individual server process from all link managers, such as a ACS subsystem process. Consequently, while a server process is processing a request, a number of requests up to the value of MAXLINKS minus 1 can queue at the server.

For COBOL servers, make sure that the value you specify in the TABLE OCCURS clause of the RECEIVE-CONTROL paragraph is greater than or equal to the number specified for MAXLINKS. If the MAXLINKS value exceeds the value in the TABLE OCCURS clause, the PATHMON process continues to grant links to the server process even after the server process has exceeded the link limit. When a link manager tries to open such a server process, the server process rejects the link.

If you omit this attribute, the default is an unlimited number of links. It is strongly recommended that you set a value for MAXLINKS.

LINKDEPTH

The LINKDEPTH attribute specifies the maximum number of links that any one link manager can have to an individual server process in a server class. (Remember, too, that a link manager can also have links to other server processes in the class.)

Unless a server process is able to process more than one request concurrently, LINKDEPTH must be set to 1. (Typically, a COBOL server has a LINKDEPTH value of 1. A communication-based server, such as an HLS server, can support a LINKDEPTH greater than 1.)

If LINKDEPTH is 1, each link manager can have only one link to each server process. Each request to the PATHMON process for an additional link results in a link to a different server process. Each link results in a single open of the server. The link manager makes the server open available for use by multiple SCREEN COBOL programs or Pathsend processes and user applications, respectively.

NOTE: Normally, only server programs that are coded as NOWAIT I/O must have a LINKDEPTH value greater than 1. Setting the LINKDEPTH value greater than 1 for a WAITED server class can lead to queuing at the server process and increased response time.

NUMSTATIC

The NUMSTATIC attribute specifies the maximum number of static servers within a server class. Because it is recommended that you run your application with static servers only, you must specify the same value for NUMSTATIC that you specify for MAXSERVERS. This way, no dynamic servers should ever be required. If you use this approach, you can the CREATEDELAY and DELETEDELAY attributes to default, because they apply only to dynamic servers.

The value of MAXSERVERS minus the value of NUMSTATIC equals the number of dynamic server processes in the server class. Because the default NUMSTATIC value is zero, failing to supply a value for NUMSTATIC causes all server processes to be dynamic. A configuration running only dynamic server processes is not recommended because the PATHMON process does not apply load balancing algorithms to dynamic server processes. Without load balancing, the first server process that starts up could be overloaded. Thus, it is recommended that you supply a value for NUMSTATIC.

The maximum number of static links that a link manager ACS subsystem processes can have is determined by the formula NUMSTATIC * LINKDEPTH.

CREATEDELAY and DELETEDELAY

CREATEDELAY is the length of time a send request queues before the link manager asks for another link, assuming the link manager has determined that no more static links are available (that is, only dynamic links are available).

A link manager will always ask for a link immediately if the link manager has determined that static links are still available.

The DELETEDELAY attribute specifies the amount of time that a dynamic server sits idle before the link manager returns the link to the PATHMON process.

For more information about static and dynamic servers, see **Configuring Server Classes** on page 62.

Understanding the Effects of Link Configuration

Application performance often depends on how well links between requestors and server processes are managed. If your application is configured in a way that creates requestor queuing problems or a bottleneck in server process responses, performance suffers.

Suppose, for example, that you set the server attribute LINKDEPTH equal to 20, which permits 20 concurrent requests to a server process. If transaction service time is 1 second and the server class handles requests serially, response time might be as much as 20 seconds. Thus, this LINKDEPTH setting can cause send requests to queue on the server class for unacceptable lengths of time.

As another example, suppose MAXLINKS is 1 and you also set the server attribute MAXSERVERS to 1, indicating that only one server process can run in this server class at a time. If this setting is too low, send requests might queue for the link manager because the PATHMON process cannot create another server process to satisfy the link requests. This situation is illustrated in Figure 3-5, which shows a server class that allows only one server process to run at a time. The server process can handle only one request: Request 1 from TCP-1. Other requests, from TCP-1, TCP-2, and the ACS subsystem process, are waiting to be processed. The problem is further complicated by the fact that if TCP-1's link is a static link, the TCP will not necessarily return the link automatically.

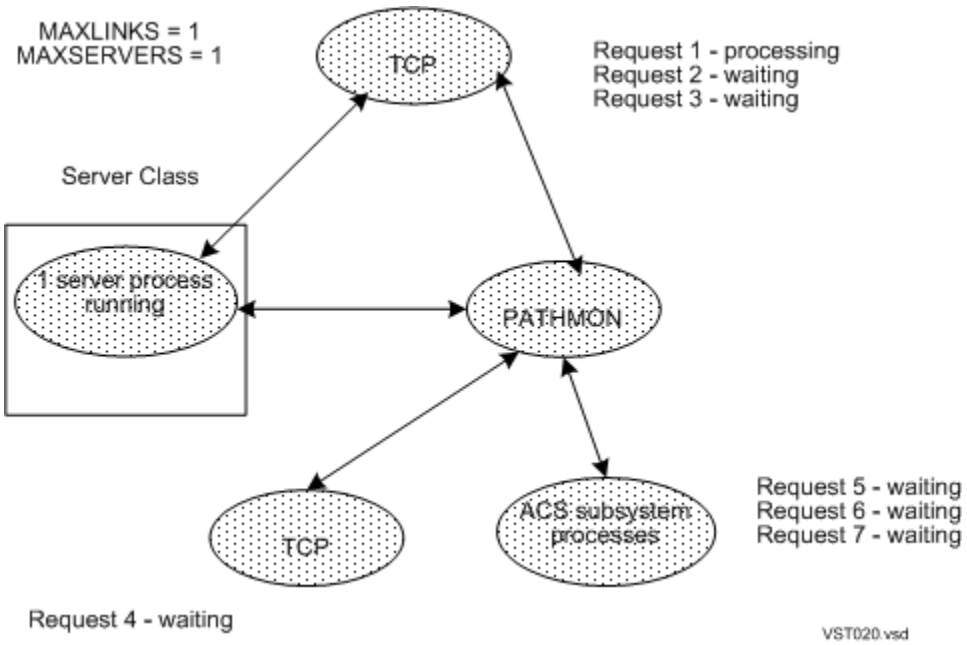

Figure 20: Link Configuration With MAXSERVERS Set to 1

Consider this configuration for a server class:

```
MAXLINKS = 5
LINKDEPTH = 1 (suggested general value)
MAXSERVERS = 1
```

This configuration specifies that a maximum of five links to each server process in this server class is allowed; each link manager can have only one link to each server process in this class; and only one server process can run in this class at any one time. Consequently, if a ACS subsystem process is managing five requestor processes, and each process sends a request to this server class, four processes must wait for the one link to become available. However, if the server configuration were changed to make MAXSERVERS = 5, each process's send request can be handled by a separate server process. Processing under this configuration is shown in **Figure 21: Link Configuration With MAXSERVERS Set to 5** on page 71.

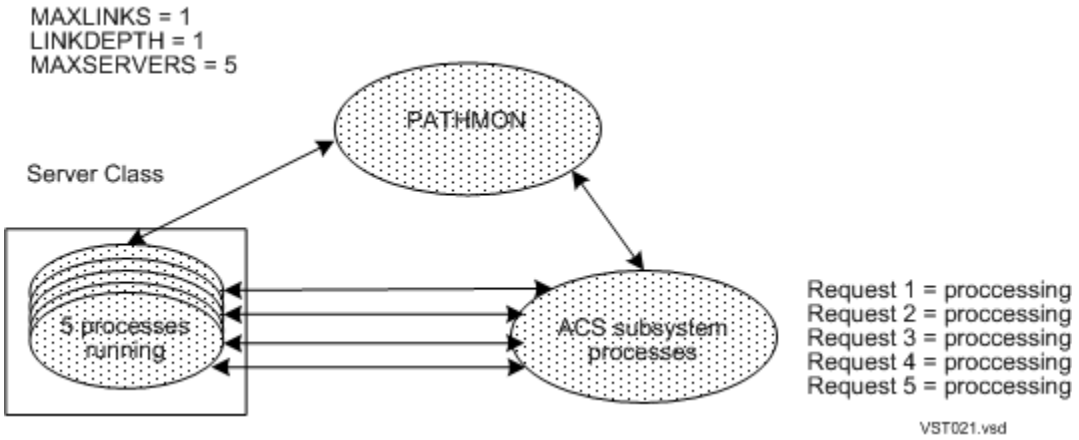

Figure 21: Link Configuration With MAXSERVERS Set to 5

Steps to Optimum Link Configuration

The balance of this subsection covers the steps you can follow to arrive at initial values for critical configuration variables that directly affect application performance. These values offer a starting point for your configuration parameters. After starting your PATHMON environment, you might want to fine tune some variables based on application performance. Guidelines are provided for performing these steps.

NOTE: Steps 1 and 3 apply to your entire configuration. Steps 4-9 apply to server classes only, and must be performed for each server class you configure. Step 10 applies to TCP configurations in Pathway/iTS environments.

---

1. Identify the number of Pathsend requestors or terminals your application must support.
2. Determine how many link managers (ACS subsystem processes, or TCPs) your configuration requires.
3. Calculate the working average transaction volume for each transaction type in the application.
4. Measure server class response time.
5. Calculate the number of active links to the server class required by each transaction type.
6. Add up the sum of all the active links for all the transaction types that use the server class to give the total active links. This is the total active links the server class must be able to support at any given moment. Use the result as the value you specify for the server class MAXSERVERS attribute and NUMSTATIC.
7. Calculate the potential link manager demand for links to the server class.
8. Calculate the value you must specify for the server class MAXLINKS attribute.

9. Adjust values based on the value in of the TABLE OCCURS clause in the RECEIVE-CONTROL paragraph of your COBOL server programs.
10. For Pathway/iTS environments, calculate additional TCP attributes.

Note that this approach is geared to generate the correct configuration values for a single server class. You need perform steps 1, 2, and 8 just once for a given system configuration; however, steps 3-7 must be repeated for each server class. When you have arrived at the values for each server class, you can total some of the values to get a basis for configuring global attributes such as PATHWAY MAXSERVERPROCESSES.

Step 1 Identify Number of Pathsend Requestors and Terminals

Identify how many Pathsend requestors and, if your environment includes Pathway/iTS, how many terminals your application environment must support. Multiply these by a safety factor—for example, 1.5—to allow for future application growth. The formula for Pathsend requestors is:

```
Pathsend requestors x constant = total Pathsend requestors
```

The formula for terminals is:

```
terminals x constant = total terminals
```

For example:

```
2000 existing terminals x 1.5 = 3000 total terminals
```

Step 2 Determine Total Link Manager Processes (LINKMON/ACS subsystem processes) and TCPs Required

Your application has one link manager process per processor. If your environment includes Pathway/iTS, a good rule of thumb is to configure one TCP for every 50 terminals. For example:

```
6 CPUs x 1 LINKMON per CPU = 6 LINKMON processes

 3000 terminals
-----------------
 50 terminals per TCP   = 60 TCPs
```

Step 3 Calculate Working Average Transaction Volume

Identify the types of transactions the application is to handle, such as inquiries, withdrawals, or summing transactions. For each transaction type, estimate the average transaction volume, or arrival rate, and multiply it by a safety factor (constant)—for example, 1.5—to allow for fluctuations in transaction volume. The result is the working average transaction volume in transactions per second.

The calculation must be done for each transaction; then the results are applied to each serve class. The formula is:

```
avg. transaction vol. x constant = working avg. transaction vol.
```

For example:

```
30 transactions per second x 1.5 = 45 transactions per second
(tps)
```

where “1.5” is the safety factor and the working average transaction volume is expressed in transactions per second.

NOTE: Because Steps 4-6 must be calculated on a per-server-class basis, a spreadsheet could be helpful to record and calculate the required values.

Step 4 Measure Server Class Response Time

Assuming that each server class provides one service, use the Measure product to determine server class response time.

Step 5 Calculate Active Links for Each Transaction

For each transaction type that accesses the server class, calculate the number of active links required by that transaction type.

```
working avg. transaction vol. x server class process response
time = active links
```

where **active links** is transactions of a given type in progress in the server class, **working avg. transaction vol.** is the result of your calculation in Step 3, and **server class process response time** is the result of Step 4.

In this example, “45 tps” is the working average transaction volume from Step 3, and “2 seconds” is the server class response time as measured.

```
45 tps x 2 seconds = 90
```

The transaction type requires 90 active links to the server class.

NOTE: A transaction type might access more than one server class to complete the transaction. The transaction might even access a given server class more than once. Be sure to account for all transaction accesses when making this calculation.

Step 6 Calculate Total Active Links: the MAXSERVERS attribute

Now add the active links for all the transactions that use the server class to get a figure for the total active links the server class must support. This figure is the value you specify for the MAXSERVERS attribute for the server class. MAXSERVERS specifies the number of server processes that must be able to run in the server class at the same time to satisfy potential demand for active links.

For example, suppose transaction-1 requires 90 active links to the server class, transaction-2 requires 5 active links, transaction-3 requires 10 active links, and transaction-4 requires 15 active links.

90 + 5 + 10 + 15 = 120 total active links required

The value you specify for MAXSERVERS must be 120.

NOTE: Because it is recommended that you run your application with static servers only, you must specify the same value for NUMSTATIC that you specify for MAXSERVERS.

Step 7 Calculate Link Manager Demand for Links

Use the results from Steps 2 and 6 to determine how many links are required to support your link manager environment. This value is used in Step 8 to calculate the value for MAXLINKS.

LINKMON Processes

For LINKMON processes, use this formula:

$$\text{no. of links required} = \text{constant} \times \frac{\text{MAXSERVERS x no. of CPUs}}{\text{no. of CPUs}}$$

where the **constant** is a safety factor, MAXSERVERS is the result of Step 6, and `no. of CPUs` is the total number of CPUs that have LINKMON processes configured. The result, `no. of links required by LINKMON processes`, is the total potential LINKMON demand for links to the server class.

This example uses the value 1 as the constant:

$$1 \times \frac{\text{MAXSERVERS=120}}{\text{6 CPUs}} = 20$$

```
20 x 6 CPUs = 120 links required by the LINKMON processes
```

Assuming the norm of one LINKMON process per processor, this configuration supports six LINKMON processes sharing 120 links.

TCPS

For TCPs, use this formula:

$$\textbf{no. of links required} = \textbf{constant} \times \frac{\textbf{MAXSERVERS}}{\textbf{no. of TCPs}} \times \textbf{no. of TCPs}$$

where the **constant** is a safety factor, `MAXSERVERS` is the result of Step 6, and **no. of TCPs** is the total number of TCPs from Step 2. The result, **no. of links required by TCPs** , is the total potential TCP demand for links to the server class.

This example uses the value 1 as the constant:

$$1 \times \frac{\text{MAXSERVERS=120}}{\text{60 TCPs}} = 2$$

```
2 x 60 TCPs = 120 links required by the TCPs
```

If your application environment is designed for a homogeneous distribution of transactions across Pathsend requestors or TCPs, you could set the value of **constant** at 1. However, to account for possible processor failure, consider using this formula:

$$1 + \frac{1}{\text{no. of CPUs - 1}}$$

If your application is designed for uneven distribution of transactions across Pathsend requestors or TCPs, increase the value of **constant** to support a potential increase in requests to a given link manager.

The result of multiplying the constant by (**`MAXSERVERS/no. of CPUs or TCPs`**) is rounded up to the next whole integer to avoid the possibility of link manager (LINKMON/ACS subsystem) processes or TCPs attempting to share fractions of links.

Step 8 Calculate the MAXLINKS Attribute

This step produces values for the key server class attribute MAXLINKS. Together, MAXLINKS and MAXSERVERS (from Step 4) provide for optimum link configuration and help maximize server class performance. In this formula, you insert values for two variables—**`no. of links required`** and MAXSERVERS—based on the results of previous steps, and then resolve the equation to arrive at a value for the third variable, MAXLINKS. Following is the formula:

MAXLINKS =	**no. of links required**---------->	from formula in Step 7
	MAXSERVERS ---------->	from formula in Step 6

where **no. of links required** is the number of links required by the ACS subsystem processes or TCPs (from Step 7) and MAXSERVERS is the requirement for total active links (from Step 6).

Resolve the equation to get MAXLINKS.

In this example, the value for **`no. of links required`** is 120, from the TCP example in Step 7. MAXSERVERS is also 120, the required number of active links.

2 =	120 TCP links required	----------> from TCP example in Step 7
	MAXSERVERS=120	----------> from example in Step 6

Resolving the equation yields a MAXLINKS value of 1, which means that in this sample configuration, each server process can have just one link to a link manager at a given moment; and 120 server processes can run at any one time. Thus, the configuration provides for a maximum of 120 links, adequate to support the 120 links required.

Note that if common sense tells you that the value of MAXLINKS is too high in relation to the value of MAXSERVERS, you must increase the value of MAXSERVERS and allow MAXLINKS to decrease. Consider a hypothetical configuration including these values:

250 =	500 TCP links required
	MAXSERVERS=2

This hypothetical configuration works on paper, but in a production application, up to 250 requestor processes could be attempting to share the services of 2 server processes, leading to many concurrent, outstanding requests to the server class. A rule of thumb to follow is that the value of MAXSERVERS can range from the result of the formula in Step 6 up to the total number of links required (from Step 7). For the hypothetical configuration above, this rule of thumb indicates you could consider increasing the value of MAXSERVERS to a number up to 500.

Step 9 Adjust Values for COBOL Servers

This step adjusts the values in Step 8 based on whether your application includes COBOL server classes. For COBOL server classes, the value specified in the RECEIVE-TABLE OCCURS clause must be greater than or equal to the number specified for MAXLINKS. If you find that the value for MAXLINKS is greater than the value in the RECEIVE-TABLE OCCURS clause, ignore the results of Step 8 and do these:

1. Set MAXLINKS to the RECEIVE-TABLE OCCURS value
2. Calculate MAXSERVERS using this formula:

```
MAXSERVERS =    no. of links required      ----------> from formula in Step 7
                ---------------------
                MAXLINKS
```

using values from Step 8 for MAXSERVERS and **no. of links** . Then adjust MAXSERVERS up until you get a MAXSERVERS value that, when inserted into the formula from Step 8:

```
MAXLINKS                =
                        no. of links required
                        ---------------------

                        MAXSERVERS
```

yields a value for MAXLINKS that equals the value of RECEIVE-TABLE in formula A.

Step 10 Identify Additional TCP Attributes

If your environment includes the Pathway/iTS product, you can perform this final step to identify additional key TCP attributes.

Use the value for the PATHWAY MAXSERVERCLASSES attribute as the value for the TCP MAXSERVERCLASSES attribute.

Use 50 as the value for TCP MAXTERMS (as in Step 2).

Use this formula to calculate TCP MAXSERVERPROCESSES:

```
All          server classes

TCP          MAXSERVERPROCESSES = Σ (no.of links          ---> from Step 7
                                    required)
                                 ---------------
                                 no. of TCPs             ---> from Step 2


             n=1
```

where all **server classes** refers to all the server classes in your configuration.

no. of links required is the number of links required by the TCP for this server class (from Step 5).

no. of TCPs is the total number of TCPs from Step 2.

Starting and Stopping SERVER Objects

After you configure and start your PATHMON environment and define and add SERVER objects, you issue the START command to activate each SERVER object or process.

NOTE: If your PATHMON environment includes objects provided by the Pathway/iTS product, such as TCP allows and TERM and PROGRAM objects, for information on starting and stopping these objects, see the *Pathway/iTS System Management Manual*.

When the PATHMON process runs the START command, it checks the status of the object named and then performs the operations needed to start the object.

NOTE: PDMCOM or PATHCOM can be used for starting and stopping SERVER objects in TS/MP 2.6 or later PATHMON environment. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information, see the *TS/MP 2.7 ACS Reference Manual*.

Starting SERVER Objects

To start a server class, an individual server process in a server class, or all server classes controlled by a given PATHMON process, use the START SERVER command.

There are two types of server processes: static and dynamic.

- A static server is a server process that the PATHMON process creates when you issue the START SERVER command.
- A dynamic server is a temporary server process that is started by the PATHMON process under certain circumstances. This type of server is not dependent on the START SERVER command.

For more information about static and dynamic servers and how you configure them, see **Configuring Server Classes** on page 62.

Static Server Processes

When you issue a START SERVER command, the PATHMON process starts the number of static servers defined for the server class. This number is defined by the NUMSTATIC attribute in the SET SERVER command.

(You do not have to issue the START SERVER command for the PATHMON process to create a static server process; the PATHMON process starts server processes as they are needed. However, issuing a START SERVER command provides for better startup and transaction performance. For more information, see **Link Requests**.)

As an example, if the value for the NUMSTATIC attribute for the server class ORDERSRV is 5, this command starts five static server processes in the server class named ORDER-SRV:

```
= START SERVER ORDER-SRV
```

The next command starts all static server processes defined for all server classes, in alphabetic order:

```
= START SERVER *
```

You can, however, specify a single static server process, by including the PROCESS option in your command. For example, the next command starts a single static server process named $ORDS in the server class named ORDER-SRV:

```
= START SERVER ORDER-SRV, PROCESS $ORDS
```

The next command starts all static server processes defined for the named server classes:

```
= START SERVER (ORDER-SRV, MFG-SRV, SALES-SRV)
```

In response to the START SERVER command, the PATHMON process performs these startup operations for each static server process in the server class:

- Creates and starts the server process by running the program specified in the PROGRAM attribute of the SERVER definition, using all parameters previously indicated by the SET SERVER commands.
- Sends a startup message to the server process, describing that process' environment. This message indicates the names of the input and output files and the default volume and subvolume used by the process, and any optional message text.
- Passes any variables or parameter assignments to the server process: ASSIGN and PARAM assignments for Guardian server processes; and *ARGLIST* and *ENV* parameters for OSS servers.

NOTE: To start a server that is frozen (with the FREEZE command) and then stopped, you must first issue the THAW command. For more information, see **Stopping SERVER Objects** on page 79.

The exact server startup protocol is as follows:

1. The PATHMON process creates the server by calling the NEWPROCESS or PROCESS_CREATE_ procedure.

   The server process context includes all DEFINEs specified in the server class configuration (using the SET SERVER DEFINE command).

2. The PATHMON process opens the servers, sends the server a startup message, and closes the server.

   The startup message has a header that is constructed from what you specified for the IN, OUT, and VOLUME attributes in the SET SERVER command. If you specified a character string for the STARTUP attribute, that is also included.

3. If the server replies with a value of 70 (FECONTINUE) to the startup message, the PATHMON process sends variables and parameter messages to the server.

The ASSIGN messages are constructed directly from the information you provided for the ASSIGN attribute to the SET SERVER command. The PARAM message contains one item for each attribute value you specified for the PARAM attribute of the SET SERVER command. The number of the processor not chosen as the primary processor is reported in the BACKUPCPU attribute, which the PATHMON process adds to the PARAM message. For a description of how this value is chosen, see the information about CPUS attribute of the SET SERVER command.

When starting a server class, the PATHMON process repeats the server startup operations for the total number of static server processes defined by the SET SERVER NUMSTATIC attribute.

Dynamic Server Processes

A dynamic server is a server process that the PATHMON process creates when a link manager asks for a link and no links to static servers are available. Note that if the link manager has determined that no more static links are available, it asks for a new link only after send requests have queued for the time period specified by the CREATEDELAY attribute for the server class.

A link manager always tries to satisfy a send request with a static link if one is available. If a dynamic server process sits idle for the time limit defined by the DELETEDELAY attribute for the server class, the link manager returns the link to the PATHMON process.

If high system performance is required, it is recommended that you not use dynamic server processes because they incur a startup penalty as part of the transaction path.

You determine the number of dynamic servers for a server class with the SET SERVER MAXSERVERS and SET SERVER NUMSTATIC attributes. If MAXSERVERS is 4 and NUMSTATIC is 2, this server class can have two dynamic servers.

Link Requests

If the PATHMON process gets a link request from a link manager to a server class, and the server class is not frozen but no server processes are started, then the PATHMON process starts only the server process that satisfies the link manager request; it does not start all of the static server processes in the server class.

Freezing and Thawing Server Classes

To disable communication between link managers and server processes without actually stopping the server class, use the `FREEZE SERVER` command. The `FREEZE SERVER` command prevents link managers from sending requests to the server processes.

For example, this command freezes all server processes in the server class SALES:

```
= FREEZE SERVER SALES
```

If no server process in the server class has incomplete or outstanding requests, the PATHMON process freezes the server class immediately. If any server process in the server class has incomplete or outstanding requests, the PATHMON process does not freeze the server class until these requests are completed. While awaiting completion of such requests, the server class remains in the FREEZE-PENDING state and can accept no new requests.

You can tell the PATHMON process to ignore the STOPMODE register, even if it is set, by terminating the `FREEZE SERVER` command with an exclamation point, as shown in this example:

```
= FREEZE SERVER SALES!
```

Keep in mind that some outstanding requests might not be fulfilled. (If the TMF subsystem is running, transactions in process are backed out as part of the freeze. For more details, see the description of the `FREEZE SERVER` command in **SERVER Commands** on page 194.)

To allow link managers to resume communication with a server class, enter the `THAW` command. For instance, to thaw the SERVER named SALES, enter:

```
= THAW SERVER SALES
```

Note that the `THAW` command works only for server classes that are frozen. The `THAW` command does not work for server classes that are in a FREEZE-PENDING state.

Stopping SERVER Objects

Before you can stop a server with the `STOP` command, you must disable all communication between link managers, such as ACS subsystem processes, or TCPs, and the server by using the `FREEZE SERVER` command.

For example, this command freezes all server processes in the server class CLASS-1:

```
= FREEZE SERVER CLASS-1
```

After the server class is frozen, you can stop the server process by stopping the server class to which it belongs. (Note that there is a short period of time during which the server classes are in a FREEZE-PENDING state and the `STOP` command will not work.) This command stops all instances of a server in server class CLASS-1:

```
= STOP SERVER CLASS-1
```

To stop all servers in your PATHMON configuration, first run:

```
= FREEZE SERVER *
```

Once the server classes are frozen, run this command:

```
= STOP SERVER *
```

After this, if the `WAIT user` option is specified, PATHMON waits until the WAIT time is over. The PATHMON process then stops these servers using the Guardian procedure call Process_Stop_.

When you enter a request to stop a server class:

1. The PATHMON process notifies all link managers, such as ACS subsystem processes and TCPs as well as all external TCPs, to delink from the server class.
2. The link managers close all server processes.
3. The server processes respond by preparing for termination and then stopping themselves.
4. For any server class that has never been linked to a ACS subsystem processes, or TCP, the PATHMON process calls the Guardian OPEN/CLOSE procedure to try and trigger the server class to terminate.
5. The PATHMON performs required clean up operations based on stop messages it receives.
6. While the PATHMON process waits to be notified that all processes are stopped, it returns a message indicating that close commands are still pending for the running processes. When the close requests are complete, the server class goes to the stopped state.

NOTE:

A PATHMON process stops only servers under its control. If a server process starts another process not under the control of a PATHMON process, the second process is not stopped when the PATHMON process stops the first server process. For example, if an OSS server runs another process, the executed process is outside the PATHMON environment. When the PATHMON process stops the OSS server, the executed process might become an “orphan” process.

To restart a stopped server process, you must first thaw the server process, then issue the `START` command as shown in this example:

```
= THAW SERVER *
= START SERVER *
```

Maintaining a PATHMON Environment

System Maintenance Tasks

This section describes operations you need to perform on a regular basis to maintain a PATHMON environment. Tasks you must perform regularly include:

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

At times, you might also need to perform other operations, such as:

- Migrate your environment to a different system
- Send messages, ranging from informative text to urgent requests, to users
- Change the owner and security of your system

Displaying Information About a PATHMON Environment

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

NOTE: PDMCOM or PATHCOM can be used for displaying information about a TS/MP 2.6 or later PATHMON Environment. However, it is recommended to use PDMCOM (instead of PATHCOM) because it can communicate with multiple PATHMONs simultaneously. For more information, see the *TS/MP 2.7 ACS Reference Manual*.

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 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 for SERVER resources.

NOTE: Information about TCPs that appears in this section applies to your environment only if you are using the Pathway/iTS product.

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. (See **Capturing a Configuration** on page 98.)

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

This scenario illustrates how to use the information provided by the STATUS commands to detect problems within your environment and determine how to rectify these problems.

A System Management Scenario

You know that users are encountering difficulties with some of their workstation client applications. The applications will not come up. By asking a few questions, you determine that RSC is running. Investigation on the host side reveals no telecommunications outages, so the problems must be in the PATHMON environment.

1. You use the STATUS SERVER * command to determine which server classes are currently running.
2. You notice that no server processes are running in the server class A-NAMESERVE. You know that this server class is required for all client applications.

   You also know that all other server classes must be stopped before the A-NAMESERVE server class can be started.

3. You decide to gracefully shut down existing applications using the FREEZE SERVER * command.
4. You use the STATUS SERVER * command to confirm that all server classes are frozen.

   The STATUS command display reveals that two server classes are still running, presumably because they contain servers processes that are still completing long transactions. These two server classes are named B-NAME-SERVE and CNAME- SERVE.

5. You use these two commands to attempt to freeze the two server classes:

   ```
   = FREEZE SERVER B-NAME-SERVER !
   = FREEZE SERVER C-NAME-SERVER !
   ```

6. You use a STOP SERVER * command to stop all server classes.
7. You use the STATUS SERVER * command again to confirm that all server classes are stopped. Now B-NAME-SERVE and C-NAME-SERVE are stopped, as are all the other server classes.
8. You use the START SERVER command to start server class A-NAME-SERVE.
9. You use the THAW SERVER * command to thaw the other server classes, followed by another STATUS SERVER * command to confirm that all these server classes are thawed.
10. You use the START SERVER * command, followed by another STATUS SERVER * command to confirm that all the server classes have started.

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 global configuration information. You can display configuration information about your overall PATHMON environment or about specific objects−the PATHMON process and SERVER objects. You can also display information about specific messages sent to user terminals through the TELL command.

NOTE:

If your environment includes the Pathway/iTS product, you can also use INFO to display information about TERM, TCP, and PROGRAM objects. For information about using the `INFO` command with Pathway/iTS objects, see the *Pathway/iTS System Management Manual*.

The INFO PATHWAY Command

The `INFO PATHWAY` command displays configuration information about your overall PATHMON environment:

```
= INFO PATHWAY
```

INFO PATHWAY Display on page 83 shows an example of the configuration information displayed when you run the `INFO PATHWAY` command.

INFO PATHWAY Display

```
PATHWAY
MAXASSIGNS 200          [CURRENTLY 63]
MAXDEFINES 0            [CURRENTLY 0]
MAXEXTERNALTCPS 0       [CURRENTLY 0]
MAXLINKMONS 8           [CURRENTLY 1]
MAXPARAMS 50            [CURRENTLY 29]
MAXPATHCOMS 95          [CURRENTLY 3]
MAXPROGRAMS 0           [CURRENTLY 0]
MAXSERVERCLASSES 50     [CURRENTLY 38]
MAXSERVERPROCESSES 300 [CURRENTLY 136]
MAXSPI 10               [CURRENTLY 1]
MAXSTARTUPS 50          [CURRENTLY 0]
MAXTCPS 0               [CURRENTLY 0]
MAXTELLQUEUE 0          [CURRENTLY 0]
MAXTELLS 0              [CURRENTLY 0]
MAXTERMS 0              [CURRENTLY 0]
MAXTMFRESTARTS 0        [CURRENTLY 0]
NODEINDEPENDENT OFF
OWNER \SYS.30,1
SCLONGPROCESSNAME ON
SECURITY "O"
```

The `INFO PATHWAY` command displays both the maximum values defined and the values currently in effect.

For example, the display in **INFO PATHWAY Display** on page 83 shows that a maximum of 50 server classes can be added to the PATHMON configuration file for this PATHMON environment. A maximum of 300 server processes can be defined for all server classes. Up to 8 ACS subsystem processes can communicate with the PATHMON process at the same time. These numbers indicate that limits for this PATHMON configuration are specified to allow for system growth.

The display also shows that a maximum of 50 server classes can have STARTUP messages; a maximum of 50 server classes can have PARAM messages, and so on. The security setting “O” prevents alterations of the configuration parameters except by the owner (\SYS.30,1).

Some of the attributes displayed by the `INFO PATHWAY` command apply only to objects configured in the Pathway/iTS environment, as follows:

```
MAXEXTERNALTCPS MAXTELLQUEUE
MAXPROGRAMS     MAXTELLS
```

```
MAXTCPS          MAXTMFRESTARTS
MAXTERMS
```

If Pathway/iTS is not installed at your site, these values must be set to zero. For instructions on setting values for these attributes, see **PATHMON Environment Control Commands** on page 159. For syntax descriptions for Pathway/iTS attributes, see the *Pathway/iTS System Management Manual*.

The INFO PATHMON Command

The `INFO PATHMON` command displays information about the PATHMON process:

```
= INFO PATHMON
```

INFO PATHMON Display on page 84 shows an example of what might be displayed when you run the `INFO PATHMON` command.

INFO PATHMON Display

```
PATHMON
  BACKUPCPU 4
  DUMP ON (FILE \SYS.$VOL1.TESTING.MONDUMP)
```

The display in **INFO PATHMON Display** on page 84 indicates that the PATHMON backup process is running in processor 4 and that the PATHMON process, if it encounters an internal error, writes the data stack information to \SYS.$VOL1.TESTING.MONDUMP.

The INFO SERVER Command

The `INFO SERVER` command displays information about a single server class, multiple server classes, or all server classes controlled by the PATHMON process.

This example displays information about a single Guardian server class, named TRECV:

```
= INFO SERVER TRECV
```

INFO SERVER Display for a Guardian Server on page 84 shows the information displayed in response to the command.

INFO SERVER Display for a Guardian Server

```
SERVER TRECV
  PROCESSTYPE GUARDIAN
  ASSIGN TEXTSRV,\SYS.$ZTXT
  ASSIGN TRAFFICLOG,\SYS.$M9.TMAILDBA.TRAFLOG
  AUTORESTART 3
  CPUS (11:4,10:5,9:6,8:7,7:8,8:9)
  CREATEDELAY 15 SECS
  DEBUG OFF
  DELETEDELAY 60 MINS
  HIGHPIN OFF
  HOMETERM $OSP
  LINKDEPTH 1
  MAXLINKS 1
  MAXSERVERS 4
  NUMSTATIC 2
  OUT \SYS.$TLOG
  OWNER \SYS.30,1
  PARAM DEBUGLOGFORMAT "TRUE"
  PARAM DEBUGLOGLEVEL "3"
  PARAM DEBUGLOGRECSPEROPEN "500"
  PARAM NAMESPACE "$T.CORR"
```

```
PARAM LOGRECSPEROPEN "10000"
PARAM COUNTMESSAGES "BOTH"
PRI 140
PROCESS $ZTR0
PROCESS $ZTR1
PROCESS $ZTR2
PROCESS $ZTR3
PROGRAM \SYS.$OPER.TRAND20.TRECV
SECURITY "N"
TMF ON
VOLUME \SYS.$OPER.TRAND20
```

This is a Guardian server class (indicated by the *PROCESSTYPE* parameter).

The object file for server class TRECV resides at \SYS.$OPER.TRAND20.TRECV (indicated by the *PROGRAM* parameter).

Server processes associated with this server class are $ZTR0, $ZTR1, $ZTR2, and $ZTR3 (indicated by the *PROCESS* parameter); these processes cannot run at a high PIN (indicated by the *HIGHPIN* parameter). A maximum of four server processes can run concurrently (indicated by the *MAXSERVERS* parameter).

Link managers, such as the ACS subsystem processes, can each have only one link to each server process within this class (indicated by the *LINKDEPTH* parameter). The maximum number of concurrent send operations to any server process in this class is one (indicated by the *MAXLINKS* parameter).

This example displays information about a single OSS server class, TRANS-SRV1, using the OBEYFORM option. OBEYFORM causes the information to be displayed in the format used to configure the server class:

```
= INFO SERVER TRANS-SRV1, OBEYFORM
```

INFO SERVER OBEYFORM Display for an OSS Server on page 85 shows the information displayed in response to the command. Note that PATHCOM includes the `RESET SERVER` command before and the `ADD SERVER` command after the server class description.

INFO SERVER OBEYFORM Display for an OSS Server

```
RESET SERVER
SET SERVER PROCESSTYPE OSS
SET SERVER ARGLIST -l, process/log, -t
SET SERVER AUTORESTART 4
SET SERVER CPUS (0,1,2)
SET SERVER CREATEDELAY 15 SECS
SET SERVER CWD /nyc/trans
SET SERVER DEBUG OFF
SET SERVER DEFINE =ACCESS, CLASS MAP, FILE $DATA.REG2.ACCESS
SET SERVER DEFINE =CONTROL,CLASS MAP, FILE $DATA.REG2.CONTROL
SET SERVER DELETEDELAY 60 MINS
SET SERVER ENV DEBUGLOGFORMAT=TRUE
SET SERVER ENV DEBUGLOGLEVEL=3
SET SERVER ENV DEBUGLOGRECSPEROPEN=500
SET SERVER ENV HANGAROUND=FALSE
SET SERVER HIGHPIN OFF
SET SERVER HOMETERM $RM123
SET SERVER LINKDEPTH 1
SET SERVER MAXSERVERS 5
SET SERVER NUMSTATIC 4
SET SERVER OWNER 101,1
```

```
  SET SERVER PRI 134
  SET SERVER PROGRAM bin/servers/srvr1
  SET SERVER SECURITY "N"
  SET SERVER STDERR process/error
  SET SERVER STDIN new
  SET SERVER STDOUT out
  SET SERVER TIMEOUT 10 MINS
  SET SERVER TMF ON
ADD SERVER TRANS-SRV1
```

This is an OSS server class (indicated by the *PROCESSTYPE* parameter).

The current working directory for the server class is /nyc/trans (indicated by the *CWD* parameter).

The object file for server class TRANS-SRV1 resides at /nyc/trans/bin/servers/srvr1 (indicated by the *PROGRAM* parameter).

Server processes associated with this server class cannot run at a high PIN (indicated by the *HIGHPIN* parameter). A maximum of five server processes can run concurrently (indicated by the *MAXSERVERS* parameter).

A TCP or ACS subsystem processes can have only one link to each server process within this class (indicated by the *LINKDEPTH* parameter).

For a complete description of all the SERVER attributes viewable through the `INFO SERVER` command, see **INFO SERVER Command** on page 199.

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. The `STATUS` command displays information about the PATHMON process, SERVER objects, ACS subsystem processes, and the PATHMON environment.

NOTE: If your environment includes the Pathway/iTS product, you can also use the `STATUS` command to display information about TERM and TCP objects. For information about using the `STATUS` command with Pathway/iTS objects, see the *Pathway/iTS System Management Manual*.

Through the `STATUS` command, various levels of detail can be reported. For instance, the `STATUS SERVER` command features an option for reporting the status of each server process in the server class.

The STATUS PATHWAY Command

The `STATUS PATHWAY` command displays status for your overall PATHMON environment. This command is useful for getting an overview of processes currently running:

```
= STATUS PATHWAY
```

STATUS PATHWAY Display on page 86 shows the information displayed in response to the command.

STATUS PATHWAY Display

```
PATHCOM
results:

PATHWAY -- STATE =
RUNNING

                RUNNING
```

Table Continued

```
EXTERNALTCPS    0
LINKMONS        8
PATHCOMS        10
SPI             1

                RUNNING  STOPPED  THAWED              FREEZE
                                           FROZEN      PENDING

SERVERCLASSES   34       4        38       0
                                                       0

                RUNNING  STOPPED  PENDING

SERVERPROCESS   98       38       0
ES

TCPS            10       1        0

                RUNNING  STOPPED  PENDING  SUSPENDED

TERMS           48       1        0        0
```

The display shows that there are eight ACS subsystem processes, 10 PATHCOM processes, and one SPI process currently running. (There are no external TCPs communicating with this PATHMON environment.) The display also shows the state of all server classes and server processes, and, if your environment includes Pathway/iTS, TCPs and TERM objects.

The STATUS PATHMON Command

The `STATUS PATHMON` command displays status for the PATHMON process, the PATHMON configuration file, and the logging files:

```
= STATUS PATHMON
```

STATUS PATHMON Display on page 87 shows the information displayed in response to the command.

STATUS PATHMON Display

```
PATHMON \COMM.$PMON -- STATE=RUNNING CPUS 8:0

PATHCTL         (OPEN)          $OPER.TRANCNFG.P
LOG1            (CLOSED)        ATHCTL
LOG2            (CLOSED)        $0

REQNUM          FILE            PID             PAID          WAIT
1               PATHCOM         $Y565           8,61          PROG-DONE
13              EXT TCP         \SYS2.$ZCMC     30,1          PROG-DONE
14              PATHCOM         \SYS.03,170     101,221
15              LINKMON         $ZL11           30,1
16              PATHCOM         $Z187           101,240
42              SPI             $UBIQ           30,1
```

The PATHMON process $PMON is running on node \COMM; the PATHCTL file, $OPER.TRANCNFG.PATHCTL, is open, but the log files are closed.

The numbers under REQNUM are the PATHMON process' internal identifiers for requestors. The FILE column indicates the type of requestor: PATHCOM, external TCP, LINKMON, SPI, and so on.

For example, requestor 1 is a PATHCOM process with process ID $Y565; the processaccessor ID is 8,61. Requestor 14 is another PATHCOM process; PROG-DONE indicates that requestor 14 is waiting for a `RUN PROGRAM` command to finish.

For a complete description of the information available through the `STATUS PATHMON` command, see **STATUS PATHMON Command** on page 186.

The STATUS SERVER Command

The `STATUS SERVER` command displays status for a single server class, multiple server classes, or all server classes defined for a PATHMON environment.

This example displays status for the server class named PROCESS-SERVER:

```
= STATUS SERVER PROCESS-SERVER
```

STATUS SERVER Display on page 88 shows the information displayed in response to the command.

STATUS SERVER Display

SERVER	#RUNNING	ERROR	INFO
PROCESS-SERVER	4		

PROCESS-SERVER currently has four server processes running.

For each server class, you can also view the status of the server processes within the server class by including the `DETAIL` option in your command, as shown in this example:

```
= STATUS SERVER PROCESS-SERVER, DETAIL
```

STATUS SERVER With DETAIL Option on page 88 shows the status for PROCESS-SERVER, a Guardian server class, and all server processes associated with this class.

STATUS SERVER With DETAIL Option

	#RUNNING	ERROR	INFO	#LINKS	WEIGHT
SERVER	4	ERROR	INFO	5	11
PROCESS-SERVER	STATE			4	10
PROCESS	RUNNING			4	9
$ZTP0	RUNNING			1	3
$ZTP1	RUNNING			0	0
$ZTP2	RUNNING			0	0
$ZTP3	STOPPED			0	0
$ZTP4	STOPPED			0	0
$ZTP5	STOPPED			0	0
$ZTP6	STOPPED			0	0
$ZTP7	STOPPED				
$ZTP8	STOPPED				
$ZTP9					

The display shows the process name for each server process. For all running processes, the display shows the number of links from link managers ACS subsystem processes or TCP—to a server (indicated by the *#LINKS* parameter). For example, $ZTP0 has five links to link managers. WEIGHT is an indication of the relative usage of the server among other servers in the server class. When a server is STOPPED, it has no links to a link manager.

If you include the `PROCESSES` option in your command, you can also view the ACS subsystem or TCP processes that currently have links to a server process, for example:

```
= STATUS SERVER PROCESS-SERVER, PROCESSES
```

You can use this command to determine if links are evenly allocated and balanced among server processes.

STATUS SERVER With PROCESSES Option: Guardian Server on page 89 shows links for a Guardian server class PROCESS-SERVER that runs in an environment that includes the Pathway/iTS product. Notice that process $ZTP2 has four links: three links with the TCP named M6530-TCP1 and one link with the ACS subsystem process named L\SYS.$ZL11. Server process $ZTP3 has only one link, with the TCP named M6530-TCP.

STATUS SERVER With PROCESSES Option: Guardian Server

```
SERVER        #RUNNING     ERROR        INFO
PROCESS-      4            3115         34
SERVER

PROCESS       STATE        ERROR        INFO         #LINKS       WEIGHT
$ZTP0         RUNNING                                5            11

LINKER        LINK COUNT
M6530-TCP2    003
M6530-TCP6    002

PROCESS       STATE        ERROR        INFO         #LINKS       WEIGHT
$ZTP1         RUNNING      3115         34           4            10

LINKER        LINK COUNT
M6530-TCP4    002
M6530-TCP5    002

PROCESS       STATE         ERROR       INFO         #LINKS       WEIGHT
$ZTP2         RUNNING                                4            9

LINKER        LINK COUNT
M6530-TCP1    003
L\SYS.$ZL11   001

PROCESS       STATE        ERROR        INFO         #LINKS       WEIGHT
$ZTP3         RUNNING

LINKER        LINK COUNT
M6530-TCP3    001

PROCESS       STATE        ERROR        INFO         #LINKS       WEIGHT
$ZTP4         STOPPED                                0            0

PROCESS       STATE        ERROR        INFO         #LINKS       WEIGHT
$ZTP5         STOPPED                                0            0

PROCESS       STATE        ERROR        INFO         #LINKS       WEIGHT
$ZTP6         STOPPED                                0            0

PROCESS       STATE         ERROR       INFO         #LINKS       WEIGHT
$ZTP7         STOPPED                                0            0
```

Table Continued

```
PROCESS      STATE          ERROR       INFO          #LINKS       WEIGHT
$ZTP8        STOPPED                                  0            0

PROCESS      STATE        ERROR         INFO          #LINKS       WEIGHT
$ZTP9        STOPPED                                  0            0
```

STATUS SERVER With PROCESSES Option: OSS Server on page 90 shows links for an OSS server class TRAN-SRVR. Notice that process $ZTX2 has two links: one link with the ACS subsystem process named L\SYS.$ZL15 and one link with ACS subsystem process L\SYS.$ZL10.

For a complete description of the information available through the `STATUS SERVER` command, see **STATS SERVER Command** on page 233.

STATUS SERVER With PROCESSES Option: OSS Server

```
SERVER       #RUNNING     ERROR         INFO
TRAN-SRVR    4

PROCESS      STATE        ERROR         INFO          #LINKS       WEIGHT
$ZTX0        RUNNING                                  5            11

LINKER       LINK COUNT
L\SYS.$ZL12  003
L\SYS.$ZL13  002

PROCESS      STATE        ERROR         INFO          #LINKS       WEIGHT
$ZTX1        RUNNING                                  3            08

LINKER       LINK COUNT
L\SYS.$ZL14  003

PROCESS      STATE        ERROR         INFO          #LINKS       WEIGHT
$ZTX2        RUNNING                                  2            05

LINKER       LINK COUNT
L\SYS.$ZL15  001
L\SYS.$ZL10  001

PROCESS      STATE        ERROR         INFO          #LINKS       WEIGHT

$ZTX3        RUNNING                                  6            12

LINKER       LINK COUNT
L\SYS.$ZL08  003
L\SYS.$ZL09  003

PROCESS      STATE          ERROR       INFO          #LINKS       WEIGHT
$ZTX4        RUNNING                                  3            07

LINKER       LINK COUNT
L\SYS.$ZL01  001
L\SYS.$ZL02  002

PROCESS      STATE          ERROR       INFO          #LINKS       WEIGHT
$ZTX6        STOPPED                                  0            0
```

Displaying Statistics Information

Server statistics are collected by ACS subsystem processes or TCPs that have links to the server class. The ACS subsystem processes collect these statistics on an ongoing basis. A TCP collects them only if you use one of these commands to direct the system to collect them:

- SET TCP STATS
- CONTROL TCP STATS

Note that TCPs are configured in your environment only if you are running the Pathway/iTS product.

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 allows 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 SERVER Command

The `STATS SERVER` command displays statistics, including response time information, about server class operations for a single server class, multiple server classes, or all server classes defined for a PATHMON environment. These statistics are gathered by the link manager that is the ACS subsystem process, that have links to the server class. This example displays statistics provided about the server class PROCESS-SERVER, a Guardian server:

```
= STATS SERVER PROCESS-SERVER
```

STATS SERVER Display: Guardian Server on page 92 shows the information collected by four TCPs and one ACS subsystem process and displayed in response to the command.

NOTE:

Your environment includes TCPs only if you have the Pathway/iTS product.

STATS SERVER Display: Guardian Server

```
SERVER PROCESS-SERVER

IN TCP                                                  07 APR 1994,
M6530-TCP1
09:47:00

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         99198        418          184          99198
REPLY                                  1744         228

RESPONSE TIME INFO (TIME VALUES IN SECS):
SEND TO SERVERCLASS

SUMMARY      # MEAS       AVG RESP     MAX RESP     MIN RESP     STAND DEV
             99195        0.12         20.05        0.00         0.26

IN TCP                                                          07 APR
M6530-TCP2                                                      1994,
09:47:00

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         1            50           50           1
REPLY                     320          320

RESPONSE TIME INFO (TIME VALUES IN SECS):

SEND TO SERVERCLASS

SUMMARY      # MEAS       AVG RESP     MAX RESP     MIN RESP     STAND DEV
             1            0.01         0.01         0.01         0.00

IN TCP M6530-TCP3                                               07 APR
09:47:00                                                        1994,

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         75           16           16           75
REPLY                     2            2

RESPONSE TIME INFO (TIME VALUES IN SECS):

SEND TO
SERVERCLASS

SUMMARY      # MEAS       AVG RESP     MAX RESP     MIN RESP     STAND DEV
             75           7.46         60.97        2.83         12.66
```

Table Continued

```
IN TCP M6530-TCP4                                    07 APR 1994,
09:47:00


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         65277        850          61           65277
REPLY                     42           42

RESPONSE TIME INFO (TIME VALUES IN SECS):

SEND TO
SERVERCLASS

SUMMARY      # MEAS       AVG RESP     MAX RESP     MIN RESP     STAND DEV
             65277        0.45         42.17        0.03         0.54

IN LINKMON                                                       07 APR
L\SYS.$ZL11                                                      1994,
09:47:01

QUEUE INFO:  REQ CNT      % WAIT       MAX WAITS    AVG WAITS    % DYNAMIC
             1            0.0          0            0.00         0.0

I/O INFO:    REQ CNT      MAX TSIZE    AVG TSIZE    I/O CNT
SEND         12           606          606          12
REPLY                     126          126
```

If you include the `FREQTABLE` option, as shown in this command example, the `STATS SERVER` command generates a frequency distribution table containing statistics for the specified server class:

```
= STATS SERVER PROCESS-SERVER, FREQTABLE
```

For more information about server statistics and the `FREQTABLE` option, see **Tuning Your System by Using Statistics** on page 120.

NOTE:

ACS subsystem processes do not report response time information or frequency distribution information. Only the TCP collects this data. Thus, this information is only available if your environment includes Pathway/iTS.

This example displays statistics about the server class TRAN-SRVR, an OSS server:

```
= STATS SERVER TRAN-SRVR
```

STATS SERVER Display: OSS Server on page 93 shows the information displayed in response to the command. Notice that all the statistics are provided by ACS subsystem processes.

STATS SERVER Display: OSS Server

```
SERVER TRAN-SRVR

IN LINKMON L\SYS.$ZL11                               04 MAR 1995,
```

Table Continued

```
08:42:06

QUEUE INFO:  REQ CNT      % WAIT       MAX WAITS    AVG WAITS     % DYNAMIC

             1            0.0          0            0.00          0.0

I/O INFO:    REQ CNT      MAX TSIZE    AVG TSIZE    I/O CNT

SEND         102          600          600          13

REPLY                     119          120

IN LINKMON L\SYS.$ZL12                              MAR 1995,
04

08:42:06

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         87           55           50           25

REPLY                     226          215

IN LINKMON L\SYS.$ZL13                              MAR 1995,
04

08:42:06

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         42           128          125          12

REPLY                     433          435

IN LINKMON L\SYS.$ZL14                              04 MAR 1995,

08:42:06

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         12           587          587          69
```

Table Continued

```
REPLY                    23          23

IN LINKMON L\SYS.$ZL15                              04 MAR 1995,

08:42:06

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         1200        14400       14000        396

REPLY                    1356        1200

IN LINKMON L\SYS.$ZL10                              04 MAR 1995,

08:42:06

QUEUE INFO:  REQ CNT     % WAIT      MAX WAITS    AVG WAITS    % DYNAMIC

             1           0.0         0            0.00         0.0

I/O INFO:    REQ CNT     MAX TSIZE   AVG TSIZE    I/O CNT

SEND         15          100         100          03

REPLY                    4235        4000
```

Reconfiguring a PATHMON Environment

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
- Add, alter, or delete objects
- Change the backup CPUs and dump files for the PATHMON process
- Exchange the primary and backup CPUs for the PATHMON process
- Change the OWNER and SECURITY attributes

For example, as your system grows, you might need to increase the maximum number of server processes to satisfy the growing demand for links to servers.

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. (You can, however, capture the current configuration of your running PATHMON environment for use at cold start by using the `INFO` command with the `OBEYFORM` option.)

Specifying new limits or changing the owner and security attributes for a PATHMON environment require that you first shut down the PATHMON process and objects, make the necessary changes, then cold start the PATHMON environment.

Specifying New Limits

You can increase or decrease limits for your overall environment using the `SET PATHWAY` command. For example, these commands specify limits for the *MAXPARAMS* and *MAXSTARTUPS* parameters:

```
= SET PATHWAY MAXPARAMS 30
= SET PATHWAY MAXSTARTUPS 40
```

You cannot specify new global limits while PATHMON-controlled objects are running; you must first shut down the entire configuration, as shown in this example. After shutdown, use a `TACL` command to start the PATHMON process again. Note that when you configure global limits after a shutdown, you must specify all limits, not just the ones you are changing. Once you have specified old and new limits, use the `COLD! start` option to restart your PATHMON environment.

```
= SHUTDOWN2, MODE ORDERLY
= EXIT
> PATHMON /NAME $PM, NOWAIT/
> PATHCOM $PM
.
.(specify old limits)
.
= SET PATHWAY MAXPARAMS 30
= SET PATHWAY MAXSTARTUPS 40
= START PATHWAY COLD!
```

For more information about shutting down and starting your configuration, see **Starting and Stopping a PATHMON Environment** on page 33.

NOTE:

When specifying limits, you must 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. For more details, see **Configuring Objects in a PATHMON Environment** on page 50.

Viewing Current Limits

To determine whether or not current limits are being reached, use the `INFO PATHWAY` command. The `INFO PATHWAY` command shows the maximum number possible and the actual number in use.

For information about the maximum number possible, see **Configuration Limits and Defaults** on page 364.

Adding, Altering, and Deleting Objects

These sections describe how to add, alter, and delete SERVER, TCP, TERM, and PROGRAM objects.

NOTE:

You manage TERM, TCP, and PROGRAM objects only if your environment includes the Pathway/iTS product. For information about the Pathway/iTS product, see the *Pathway/iTS System Management Manual*.

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 must be aware of the 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 SERVER, 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 and adding objects, see **Configuring Objects in a PATHMON Environment** on page 50.

Altering Objects

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

Before using the ALTER command for a SERVER object, you must freeze and stop the SERVER with the FREEZE and STOP commands, respectively. This example changes the priority at which the server runs:

```
= FREEZE SERVER SALES
= STOP SERVER SALES
= ALTER SERVER SALES PRI 50
= THAW SERVER SALES
= START SERVER SALES
```

Deleting Objects

You can delete SERVER, TCP, 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 SERVER, you must freeze and stop the SERVER by using the FREEZE and STOP commands, respectively.

To delete the server class named SALES, you enter:

```
= FREEZE SERVER SALES
= STOP SERVER SALES
= DELETE SERVER SALES
```

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 the PATHMON process while it is running:

- The backup processor 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 processor for the PATHMON process in a multiprocessor system, you enter the CONTROL PATHMON command. For instance, to change the backup processor for the PATHMON process to processor 6, enter:

```
= CONTROL PATHMON, BACKUPCPU 6
```

In response, the PATHMON process stops the backup process and creates a new backup process in the processor that you specify.

To direct the memory dump for the PATHMON process to a file named PMDUMP, enter:

```
= CONTROL PATHMON, DUMP ON (FILE PMDUMP)
```

The PATHMON configuration file is updated to reflect these changes.

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 PATHMON process by using the SWITCH command, as shown in this example:

```
= SWITCH PATHMON
```

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

```
= PRIMARY PATHMON
```

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

Changing the Owner and Security Attributes

You can change OWNER and SECURITY attributes by using the SET PATHWAY command. For example, these commands specify that only the owner—user ID 8,60— can modify the PATHMON environment:

```
= SET PATHWAY OWNER 8,60
= SET PATHWAY SECURITY "O"
```

You cannot change the OWNER and SECURITY attributes while a PATHMON environment is running; you must first shut down the environment, as shown in this example. Recall that when you configure global limits after a shutdown, you must specify all limits, not just the ones you are changing. Once you have specified old and new limits, and changed the OWNER and SECURITY attributes, you must use the COLD! start option to restart the PATHMON environment:

```
= SHUTDOWN2, MODE ORDERLY
= OPEN $PM
.
. (specify old limits)
.
= SET PATHWAY OWNER 8,60
= SET PATHWAY SECURITY "O"
= START PATHWAY COLD!
```

For more information about the OWNER and SECURITY attributes, see **Configuring Objects in a PATHMON Environment** on page 50.

Capturing a Configuration

You can use the INFO command with the OBEYFORM option to capture a configuration for use in future start operations.

When you specify the OBEYFORM option, the PATHMON process reads information from the PATHMON configuration file and writes it to a specified file in the format required for a PATHMON environment startup file. Each attribute is written as a syntactically correct PATHCOM command with the proper SET prefix. **Using the OBEYFORM Option** on page 99 contrasts output of the INFO command with and without the OBEYFORM option specified:

Using the OBEYFORM Option

```
= INFO PATHMON
PATHMON
BACKUPCPU 4
DUMP ON (FILE \SYS.$VOL1.TESTING.MONDUMP)
= INFO PATHMON, OBEYFORM
SET PATHMON BACKUPCPU 4
SET PATHMON DUMP ON (FILE \SYS.$VOL1.TESTING.MONDUMP)
```

You can use the OBEYFORM option to create a comprehensive record of your complete PATHMON environment or, optionally, a selected portion of it. The OBEYFORM file represents a "snapshot" of the system as it exists at the time the file is produced.

These steps illustrate how to use OBEYFORM to capture a complete system configuration:

1. Create an EDIT file that contains a set of INFO commands specifying the OBEYFORM option.

   Each individual INFO command captures a distinct portion of the overall configuration (the PATHMON process, PATHWAY object, SERVERS, TCPs, and so forth) in the destination command file.

   For example, the second INFO command in this group captures the configuration information for the PATHWAY object and writes that information to the startup file named NEWCONF. Notice that an asterisk (*) is used in the last four commands to indicate all SERVER objects in the set.

   ```
   INFO /OUT NEWCONF/ PATHMON , OBEYFORM
   INFO /OUT NEWCONF/ PATHWAY , OBEYFORM
   INFO /OUT NEWCONF/ SERVER *, OBEYFORM
   ```

2. Purge any existing file of the same name as your destination command file, or rename that file, by entering the PURGE or RENAME TACL commands.

   This step prevents OBEYFORM from appending new information to a file that already contains obsolete information. For example, to purge the file named NEWCONF, enter:

   ```
   6> PURGE NEWCONF
   ```

   To rename the file named NEWCONF to OLDCONF, enter:

   ```
   7> RENAME NEWCONF, OLDCONF
   ```

3. Produce the OBEYFORM file. To do this, enter a PATHCOM OBEY command, specifying as input the EDIT file described in Step 1. For example, if you had named the EDIT file INFOCOM, you would enter:

   ```
   = OBEY INFOCOM
   ```

   This OBEY command runs the INFO commands contained in the EDIT file. With the OBEYFORM option, an INFO command generates configuration information in the form of RESET, SET, and ADD commands that can be run by PATHCOM.

4. Shut down your PATHMON environment, using the SHUTDOWN2 command as explained in **Shutting Down a PATHMON Environment** on page 43.

5. Alter any object attribute in the OBEYFORM file to conform to your requirements.

6. Add the START PATHWAY command to the command file, just after the last SET PATHWAY command. OBEYFORM does not automatically write the START PATHWAY command to its destination file, so you

must include it before using this file to start a PATHMON environment. For example, in this command file, you would insert START PATHWAY COLD! in the location indicated:

```
.
.
SET PATHWAY MAXTMFRESTARTS 5
SET PATHWAY OWNER \QATST.8,21
SET PATHWAY SECURITY "N"
START PATHWAY COLD! <----(Required for COLDSTART)
.
.
```

7. Run the commands in the command file by entering the `OBEY` command, specifying the command file as input. As PATHCOM runs these commands, it configures and cold starts the PATHMON environment. As an example, to use the command file named NEWCONF to configure and start your system, enter:

   ```
   = OBEY NEWCONF
   ```

   This step completes your configuration and start operations.

If you frequently alter your global configuration, using the `OBEYFORM` option to create a command file can be helpful.

NOTE:

OBEYFORM presents certain limitations. First, it produces a highly restrictive record of your system configuration, one in which the default values recorded are those assigned by the current version of the PATHMON environment. The file names recorded are fully expanded on the basis of current file assignments at your node. In addition, OBEYFORM produces files that contain values for all object attributes, whether assigned by a PATHCOM command or by default.

Logging Status and Error Information

PATHMON reports error and status information to a log file.

You can request the PATHMON process to report errors only, or errors and changes in object status as well. You can request the PATHMON process to log information to only one file, or to copy the information to two files. Also, 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 when they scroll off the terminal screen; also, performance is generally better to a process on a disk file than to a terminal.

Security Logging

The TS/MP 2.6 or later PATHMON makes a log entry for the following commands when they are issued on the objects configured under PATHMON:

- ABORT
- ADD
- ALTER
- DELETE

- FREEZE
- STOP
- THAW
- START

The audit logging is done when the STATUS messages are d. The message format is shown below:

```
ddMMMyy,mm:ss <pathmon name>: STATUS - *1166* <entity name>,
AUDIT MSG - <command name> <command name> <entity type>
<status> - USER <user id>
```

where:

ddMMMyy,mm:ss

Displays date and time.

<pathmon name>

Displays the PATHMON process name that sends the message.

<entity name>

Displays the entity name.

<command name>

Displays the command, which is issued with these possible values: ALTER, ADD, DELETE, FREEZE, STOP, THAW, START, and ABORT.

<entity type>

Displays an entity name such as TCP, TERM, SERVERCLASS, and so on.

<status>

Displays the status as failed or successful.

<user id>

Displays the ID of the user who issued the command.

The status messages are currently logged for successful `FREEZE` and `THAW` commands. They will continue to appear in their current format. Additionally, the audit logs will also be captured.

Logging Information to a Terminal

The initial log file is the OUT file for the PATHMON process. In interactive mode, this is typically your command terminal.

When you use your command terminal for logging, you can get immediate information from the PATHMON process. You must restrict the activity on that terminal, however, to avoid interfering with the PATHMON messages; in fact, if you use a terminal for logging, it is best to have that terminal dedicated to logging.

If you are using a terminal as the logging file, type “pause” at the TACL prompt to suspend the TACL process so that logging can occur.

Logging Information to a Disk File

To select a destination other than your terminal for logging information, you can enter a run option with the implicit `RUN PATHMON` command, as follows:

```
5> PATHMON / NAME $PMX, CPU 3, NOWAIT, OUT LOGPMON/
```

Alternatively, you can specify the file for logging output by using 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
```

Before specifying a disk file for logging, you must first create that file, using the `FUP CREATE` command. (You must ensure that the file is large enough.)

The PATHMON process records the log file names that you select in the PATHMON configuration file. After a cool start, the same log files specified in the configuration file are used again; records produced after the cool start are appended to these files.

If an error occurs while error information, status information, or both are being written to a log file, the PATHMON process does:

1. Closes the log file specified with the **`file-name`** parameter.
2. Opens $0.
3. Writes the information to $0. The PATHMON process writes error information, status information, or both to $0, depending on what was specified for the initial log file with the *STATUS* parameter. Additionally, the PATHMON process writes the information to $0 in the same format (that is, event messages or text) specified for the initial log file with the *EVENTFORMAT* parameter.

If an error then occurs while information is being written to $0, the PATHMON process closes $0 and prohibits further error and status logging. The PATHMON process does not reopen either the log file or $0.

In this example, you have configured two log files:

```
= LOG1 $0, EVENTFORMAT
= LOG2 $DATA.PM.LOGFILE, STATUS
```

If an error occurs on LOG1, the PATHMON process closes the log because the error occurred on $0; the PATHMON process would not reopen LOG1. If an error occurs on LOG2, the PATHMON process reopens $DATA.PM.LOGFILE as LOG2 $0, STATUS. The PATHMON process next sends both error information and status information to $0 because although the `LOG1` command did not specify the *STATUS* parameter, the `LOG2` command did. Additionally, the PATHMON process sends both event messages and text messages to $0 because although the `LOG2` command did not specify the *EVENTFORMAT* parameter, the `LOG1` command did.

NOTE:

If the log file is running but is temporarily unavailable or overloaded, the PATHMON environment might slow down.

Links and PATHMON Performance

A link is managed and owned by the PATHMON process that controls the server process. To perform link management, PATHMON maintains status information for SERVER objects as well as ACS subsystem processes. If your environment includes the Pathway/iTS product, the PATHMON process also maintains status information for TCP and TERM objects.

By understanding how PATHMON manages links and what causes dissolution of links, you can take steps to improve the performance of your system.

This can adversely affect PATHMON performance under exception conditions such as network or processor failure:

- Maintaining object status to manage links

  To manage links, the PATHMON process maintains status information about PATHMON controlled objects and ACS subsystem processes. During exceptions, managing communication between these objects and server processes can slow down the response time of PATHMON processes compared to other commands (from PATHCOM or SPI).

- Creating and deleting links

  If the PATHMON process is configured so that it is constantly creating and deleting links, the server might wait for processor cycles, causing incoming transactions to be placed on the server-class wait queue.

For more information about PATHMON performance and steps you can take to improve it, see **Managing PATHMON Process Performance** on page 104.

Understanding the Causes of Link Dissolution

A PATHMON process, ACS subsystem processes, or TCP process, or a server process (indirectly) can delete a link. This table describe how a link manager can cause deletion of a link.

A link manager will return a link to a server process when any of these situations occur:

Cause	Action
A “DELETEDELAY” timer expires for a dynamic link.	Returns the dynamic link.
An OPEN error occurs on a link.	Returns the 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.)	Returns all links for this server process, allowing current I/O to complete.
The PATHMON process sends a server class stop request.	Returns all links for this server process, allowing current I/O to complete.
The PATHMON process sends a forced delink request.	Returns all links to the server process, allowing current I/O to complete.
The PATHMON process sends an external shutdown notification.	Deletes all links owned by that PATHMON process, allowing current I/O to complete. Consequently, no links are returned to the PATHMON process.
The PATHMON process returns an unexpected I/O error or close message, or the link manager detects a message protocol error.	Deletes all links owned by that PATHMON process, allowing current I/O to complete. Consequently, no links are returned to the PATHMON process.
A link manager gets an error trying to communicate with 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.

The PATHMON process and Server Links

If a server process goes down, the operating system automatically informs the PATHMON process that owns that server class. The PATHMON process then dissolves all links to that server process by sending a message to all ACS subsystem processes with established links to that server process.

Because servers are (usually) coded to stop themselves when all openers have closed them, system managers seldom need to worry about stopping servers.

Managing PATHMON Process Performance

The performance of the PATHMON process is critical to response time for applications and management tasks.

Generally, in the production environment, the PATHMON process must be busy only during startup and shutdown of the PATHMON environment. If the PATHMON process is very busy at times other than startup and shutdown, you must take steps to alleviate the workload.

These sections describe factors that can cause poor performance and steps you can take to improve performance.

Detecting Problems

These factors might cause the PATHMON process to be busy at times other than startup and shutdown:

- Poor configuration
- Creating and deleting links
- Gathering statistics and information
- Logging errors
- Failures

Poor Configuration

If a link manager, such as a ACS subsystem processes, or TCP, cannot establish communication with a server process, the link manager indicates this situation to the screen program or Pathsend requestor, and the PATHMON process writes an error message to the log files. If you experience such a failure, you likely need to review your configuration. You might have to increase the configuration size to handle the load, or a server class naming error might occur.

Creating and Deleting Links

If the PATHMON process is configured so that it is constantly creating and deleting links, it can become the busiest process in the processor.

The PATHMON process can also become busy creating and deleting links if an application repeatedly sends Pathsend requests to a frozen server class, and the ACS subsystem processes has no links to that server class. The ACS subsystem processes repeatedly ask the PATHMON process for a link to the frozen server class, then returns the link when the server is found to be frozen. This problem can be resolved if the application, when notified that the server class is frozen, waits a given period of time before trying again. For more information on configuring for optimum link management, see **Configuring Links for Optimum Performance** on page 67.

Gathering Statistics and Information

The processor cost to maintain statistics is low. However, repeated reporting on statistics (especially with the `DETAIL` option) or configuration information can impose an added burden on the PATHMON process and the processor.

Logging Errors

The PATHMON process is responsible for all error logging. Delays in logging errors or status messages can adversely affect the performance of the PATHMON environment.

You must review the log file and understand what causes each of the errors. In a well-designed, well-configured PATHMON environment, the log might be empty or nearly so. Occasional errors do not exact a large penalty. However, a poor configuration—for example, one that might cause processes to frequently start and stop—can cause a performance degradation just in the logging of the process status.

Failures

For large systems in which the PATHMON process manages numerous links, this events can cause performance problems:

- processor failure
- Network failures
- A TCP going down (Pathway/iTS environment only)
- Server process failure

In certain network situations, communication delays between the PATHMON process and link manager that is the ACS subsystem process can reduce response time even further.

Improving Performance

You can improve PATHMON performance in some instances by reconfiguring your PATHMON environment. These subsections describe options that you might want to consider.

Configure for Sufficient Number of Links

You must ensure that there are enough links available in the system initially by specifying appropriate values for MAXLINKS, LINKDEPTH, MAXSERVERS, and MAXSERVERPROCESSES.

Use Multiple PATHMON Processes to Manage Servers and Associated Links

You can create separate PATHMON processes to manage servers and their associated links. By distributing link management among multiple processes, you improve response time. The cost, however, is increased operational complexity.

Information to Include When Reporting Problems

When requesting your Hewlett Packard Enterprise representative's help in resolving a problem with NonStop TS/MP, you can greatly enhance the representative's ability to help you by providing this information, where applicable.

PATHMON Environment

These files and information apply to the PATHMON environment as a whole.

Version Procedure

Use the VPROC procedure to get the product version for the applicable PATHMON components, as follows:

```
>VPROC
Enter filename:
>$<vol.subvol>.PATHCOM
>$<vol.subvol>.PATHMON
```

where **$vol.subvol** is the system volume and subvolume on which the desired component resides.

Event Log and CONFLIST

Be sure to collect any PATHMON and EMS log files. Also include the $SYSTEM.SYSnn.CONFLIST file.

Configuration and Startup Information

If you are able to reproduce the problem, give the Hewlett Packard Enterprise representative a simple test case illustrating the problem along with a description of how to reproduce it. Otherwise, give the representative the object modules and data files (PATHCONF files) making up the PATHMON configuration when the error occurred.

PATHMON-Specific Problems

Collect this information when you detect a problem with the PATHMON process itself.

PATHMON Dump

As explained in **Requesting Error Dumping** section in **Configuring the PATHMON Process**, you can request error dumping by specifying either the SET PATHMON, DUMP ON command or the CONTROL PATHMON command. If you request your Hewlett Packard Enterprise representative's help in analyzing a problem, the representative will likely require a DUMP file. It is therefore recommended that DUMP be set to ON for production systems.

If error dumping has not been specified and the PATHMON tables (showing, for example, server states) appear to be in error, your representative may ask that you force a dump by performing:

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

where **pm** is the name of your PATHMON process and **pmdump** is name of the dump file.

Note that 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 internal state of the PATHMON process at a particular time—before it encounters a fatal error. You must only force a dump if your Hewlett Packard Enterprise representative specifically requests it.

PATHMON Configuration and Object Information

Make the PATHCTL file available to your Hewlett Packard Enterprise representative. Also, use PATHCOM commands to display information about PATHMON-controlled objects at the time of the problem, as follows:

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

where **pm** is the name of your PATHMON process and **object** is the name of a server class.

SPI Data

If the PATHMON problem is SPI-related, get the source code and object module of the DSM application process.

PATHCOM-Specific Problems

If detect a problem with PATHCOM, collect the following for your Hewlett Packard Enterprise representative's review:

- The PATHMON configuration file, PATHCTL
- The command sequence that produced the problem

Server-Specific Problems

Collect this information if you detect a problem with a server class:

- Server source code and object modules.
- Requestor source code and object modules.
- For pending servers, a PATHMON dump
- Status information; at the system prompt, perform:

  ```
  STATUS /OUT <statfile>/ *, PROG / *, PROG >, DETAIL
  ```

  where `statfile` is the listing file that captures the status display in an edit file, and prog filename is the server program's file name.

Recovering from PATHMON Failure

If PATHMON fails, perform these steps to capture data about the failure and restart the PATHMON process.

1. Determine whether it is in your users' best interests to bring the PATHMON environment down right away. A well-designed, well-tuned system might run for one or two days without the PATHMON process. Contact users and find out whether critical applications are running as expected. You might be able to wait before stopping the entire environment.
2. Capture data about the failure. Use VPROC to get the product version. If you are running the PATHMON environment with DUMP ON, retrieve dumps from ZZPM**nnn** . Get any relevant messages from the PATHMON log and the EMS log. Note what commands you were using or functions you were performing at the time of the failure. Collect this information to have available when you contact your Hewlett Packard Enterprise representative for help.
3. Bring the PATHMON environment down using `Guardian STOP` commands to stop all processes. For more information about the `Guardian STOP` command, see the **Guardian User's Guide**.
4. Start PATHMON again.
5. Coolstart the PATHMON environment.

Keep Development and Production Separate

Hewlett Packard Enterprise recommends that you do not mix your development and production environments. Ideally, maintain 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 Inspect product to debug the server. During this time, the server is unavailable, and any processes attempting to establish links to the server could encounter an unacceptable delay. Similarly, a TCP that handles both production and development SCREEN COBOL requestors 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 occur when you are debugging servers in a Pathway application. For further information about these errors, see the section on servers in the *TS/MP 2.5 Pathsend and Server Programming Manual*.

Maintaining Associative Server Processes

An **associative server** is a process within a server class that can be started outside of the PATHMON environment by a process other than the PATHMON process that controls the server class. An associative server process is configured with this command:

```
= SET SERVER PROCESS $process-name (ASSOCIATIVE ON)
```

The PATHMON process can change the state of an associative server process from stopped to running without actually creating a new server process. If the PATHMON process attempts to create an associative server process and discovers that the server process already exists, the PATHMON process changes the server state to running and uses the existing server process (instead of reporting an error).

Similarly, the PATHMON process can change the state of an associative server from running to stopped even if the process is still running. This situation can occur when:

- A `STOP SERVER` command is issued. The PATHMON process only waits until all links granted by the PATHMON process are returned by the ACS subsystem processes or the TCP process.
- The last link granted by the PATHMON process is returned by a link manager that is the ACS subsystem processes, however the server was not started by the PATHMON process.

In both of these cases, the PATHMON process changes the state to stopped without waiting for the server to terminate.

When the last link granted by the PATHMON process is returned by a link manager, such as the ACS subsystem processes, and the associative server is started the server state is left as running (instead of being changed to pending). When the server state is running, the PATHMON process can continue to allocate links to the server during the period that other requestors continue to use the server.

There is an interval when the PATHMON process can grant a link to an associative server process that is stopping itself. Although the stopped state allows you to use the `ALTER SERVER` command to specify changes for configuration attributes, the changes you specify do not take effect until the associative server is actually stopped and then restarted from the same the PATHMON process. Changes to these attributes do not take effect until the PATHMON process receives a `START SERVER` command for an associative server that has actually stopped:

ASSIGN	OUT	PRI
DEFINE	OWNER	PROGRAM
GUARDIAN-LIB	VOLUME	SECURITY
HOMETERM	PARAM	STARTUP
IN		

Associative Servers as Subtype 30 Processes

An associative server can be a subtype 30 process, which simulates a terminal or communication device. When an associative server is a subtype 30 process, the PATHMON process determines the server's device type; you do not have to designate the device type with the `TYPE` option of the `RUN PROGRAM` command. This configuration is typical when a server process acts as a front end process (fep) for a terminal. High-level language support (HLS) is often configured this way.

Migrating Your Environment to a Different System

Careful preparation is required to switch your PATHMON environment to a different system, either in the event of failure or as part of a planned migration.

The best preparation for any such move is to anticipate it when you first configure your PATHMON environment. When you set global attributes using the `SET PATHWAY` command, set the NODEINDEPENDENT attribute to ON to make unspecified node names for processes and devices default to the node where the PATHMON process is running after cool start. By configuring your application for node independence and then leaving the node unspecified in any name you configure, you can avoid much of the work involved in renaming objects and processes configured with hard-coded node names. For more information on configuring your system for node-independence, see **Specifying Node Independence** on page 53 and **SET PATHWAY Command** on page 171. The SET PATHWAY syntax description includes a list of objects and attributes that are node-independent when the NODEINDEPENDENT attribute is set to ON.

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, processor and user ID specifications. This table lists object attributes that you might have to change with the `ALTER` command when migrating your application. The table includes objects supported by the NonStop TS/MP product and, as noted, the Pathway/iTS product.

Table 3: Migration Considerations: Object Attribute Values

Object or Process	Attribute	Considerations/ Recommendations
PATHMON process	BACKUPCPU **`cpu-num`**	Ensure that CPUs on new system have same numbers or change manually.
	DUMP ON | OFF	Specify a new DUMP file name if the name was previously specified.
PATHWAY object	OWNER **`owner-id`**	Ensure that user ID is known to new system or change manually.
SERVER CLASS	CPUS **`cpu-list`**	Ensure that CPUs on new system have same numbers or change manually.
	DEFINE **`define-spec`**	Could include system name stored as buffer; check and change manually if required.
	HOMETERM **`termname`**	Ensure that a terminal with same name exists on new system or change manually.

Table Continued

Object or Process	Attribute	Considerations/ Recommendations
	OWNER **owner-id**	Ensure that user ID is known to new system or change manually.
	PARAM **param-string**	Could include system name stored as buffer; check and change manually if required.
	STARTUP **startupstring**	Could include system name stored as buffer; check and change manually if required.
	VOLUME **$volume**	Ensure that volume is known to new system or change manually. System name might be passed in startup message; check and change manually if required.
SERVER PROCESS	CPUS **pricpu:backupcpu or cpu-num**	Ensure that CPUs on new system have same numbers or change manually.
	GUARDIAN-SWAP **$volume**	Ensure that a disk attached to new system has same name or change manually.
	HOMETERM **termname**	Ensure that a terminal with same name exists on new system or change manually.
TCP*	CPUS **pricpu:backupcpu**	Ensure that CPUs on new system have same numbers or change manually.
	DUMP	Ensure that volume portion of the file name is known to new system or change manually.
	HOMETERM **termname**	Ensure that a terminal with same name exists on new system or change manually.
	TCL PROG	Ensure that volume portion of the file name is known to new system or change manually.
TERM*	FILE **filename**	Ensure that a device with same name exists on new system or change manually.

Table Continued

Object or Process	Attribute	Considerations/ Recommendations
	TCL PROG	Ensure that volume portion of the file name is known to new system or change manually.
PROGRAM*	OWNER **owner-id**	Ensure that user ID is known to new system or change manually.
	PRINTER **filename**	Ensure that a device with same name exists on new system or change manually.
	TCL PROG	Ensure that volume portion of the file name is known to new system or change manually.
* Objects configured under Pathway/iTS only		

NOTE: The SWAP and GUARDIAN-SWAP attributes are not included in Table 5-1 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. If the SWAP attribute is specified for TCP, the specified volume is used. If not, KMSF SWAP space is used; if this SWAP space is not sufficient, it uses the volume name defined for the TCLPROG as SWAP space. In a node-independent environment, it is recommended that you do not specify values for these attributes.

You must also consider whether Pathsend requestors or SCREEN COBOL applications —perhaps those controlled by remote systems—use hard-coded node names in requests to server processes that belong to the application you want to migrate. If Pathsend procedure calls and screen application requests have not been coded for node independence, the applicable code must be changed manually.

If you plan to switch your application to a new system, and especially if you plan to migrate your application on a regular basis, you must consider creating an Command file to accomplish any required manual changes.

Managing the Pathsend Environment

The Pathsend Environment

This section discusses the Pathsend environment, which consists of Pathsend processes and ACS subsystem processes, and the tasks you perform to manage the Pathsend environment. For more information on ACS subsystem core processes, see the *TS/MP 2.7 ACS Reference Manual*.

Pathsend Processes

Pathsend processes are user-written Guardian programs that use Pathsend procedure calls to communicate with server classes. The Pathsend procedure calls are part of the operating system library.

For more information about the Pathsend procedure calls and Pathsend processes, see the *TS/MP 2.5 Pathsend and Server Programming Manual*.

ACS Subsystem Core Processes

Together with the PATHMON process, the ACS subsystem controls communication between Pathsend requestor processes and server classes. As a benefit, the ACS Subsystem provides higher Pathsend limits and better link management over TS/MP 2.0 LINKMON process. The properties of the ACS Subsystem core processes are described in the subsections that follow.

ACS Subsystem Core Tasks

ACS subsystem core processes a multitasking environment that handles the interface to Pathway servers from requestors other than SCREEN COBOL requestors. These requestors include the Pathsend processes, such as those created by RSC on behalf of the workstation clients.

The ACS subsystem gathers statistics for servers. For more information, see **Tuning Your System by Using Statistics** on page 120.

Communication With the PATHMON Environment

The Process Broker (PB) process in the ACS subsystem communicates with the PATHMON environment only if it has a link to a server process running in that environment. When the PB process no longer has any links to any server process in a PATHMON environment, it terminates its connection to that environment.

NOTE: For more information on the PB process, see the *TS/MP ACS Reference Manual*.

Execution

The ACS subsystem supports access to server classes by requestor programs written in TAL, C, C++, COBOL, or Pascal. It supports access by assisting with the execution of Pathsend procedure calls.

Each processor contains only one set of ACS core processes (Redirector (ROUT), PB, and configuration subscriber (CS) programs running as $ZL**nn** , $ZP**nn** , and $ZCS**nn** processes respectively, where **nn** is the processor number), which are responsible for all Pathsend requestors executing in that processor. ACS subsystem core processes acquire links from the PATHMON process that controls the server class. The ROUT process then shares the links among the Pathsend requestors executing in the processor.

The ACS subsystem core processes are visible to a PATHMON process but are not controlled by it. Unlike the TS/MP 2.0 LINKMON process, you can directly configure and control the ACS subsystem using the Subsystem Control Facility (SCF) commands. You can use PATHCOM commands to perform various ACS subsystem related functions. For more information on managing the ACS subsystem and the ACS core processes, see the **TS/MP 2.7 ACS Reference Manual**.

The following table lists the effects of various PATHCOM commands on the ACS subsystem.

Table 4: PATHCOM/PDMCOM Commands That Affect the ACS Subsystem

PATHCOM/PDMCOM Command	Effect on ACS Processes
`FREEZE SERVER`	Prohibits the ACS subsystem from sending requests to server classes.
`GET-LINK`	Acquires links to a serverclass in advance without using a Pathsend request.
`INFO PATHWAY`	Displays the current MAXLINKMONS value.
`RESET SERVER SECURITY`	Resets the current values for a server class and server class security.
`ALTER SERVER SECURITY`	Alters the current values for a server class and server class security.
`SET PATHWAY`	The *MAXLINKMONS* parameter sets the maximum number of ACS PB processes that can communicate with a specific Pathway environment.
`SET SERVER SECURITY`	SECURITY parameters restrict access by Pathsend requestors to the server classes.
`SHOW SERVERINFO SERVER`	Checks the current SECURITY and OWNER attribute values for a server class.
`STATS SERVER`	Displays resource usage and performance statistics on ACS subsystem processes linked to a specific server class.
`STATUS LINKMON`	Checks the status of ACS PB processes that have links to server classes in the Pathway environment.
`STATUS PATHMON`	Identifies the ACS PB processes currently communicating with the PATHMON process.
`STATUS PATHWAY`	Determines the number of ACS PB processes communicating with a Pathway environment.
`STATUS SERVER`	Identifies the ACS PB process linked to a specific server class.
`THAW SERVER`	s ACS subsystem processes to send requests to frozen server classes.

For more information about the commands listed in **PATHCOM/PDMCOM Commands That Affect the ACS Subsystem**, see **PATHCOM Operation Commands** on page 144, **PATHMON Environment Control Commands** on page 159 and **SERVER Commands** on page 194.

If you have TS/MP 2.6 or later installed on your system, you need to configure and bring up the ACS subsystem processes using SCF commands. You cannot start ACS subsystem processes with `TACL RUN` commands.

Process Names

ACS subsystem processes appear in PATHCOM displays in the form L\system.$ZPnn, where nn is the number of the processor on which the ACS subsystem processes reside. PATHCOM adds an “L” before the system name to indicate that it is a link manager process, for example:

```
L\ABC.$ZP05
```

The following table lists the ACS core processes and their respective process names and object names.

Table 5: ACS Core Processes

ACS Core Process	Process Name	Object Name
BC	($ZACS)	$SYSTEM.SYSnn.BC
PB	($ZPnn)	$SYSTEM.SYSnn.PB
CS	($ZCSnn)	$SYSTEM.SYSnn.CS
ROUT	($ZLnn)	$SYSTEM.SYSnn.ROUT

Initialization

Unlike the TS/MP 2.0 LINKMON process, the ACS subsystem core processes are initialized as soon as they are started. The status of ACS core processes on each CPU on the system is logged in the ACS subsystem log file. The ACS subsystem log file can be set while configuring the ACS subsystem using the `SCF` commands.

Limits

The ACS subsystem provides higher Pathsend limits per CPU compared to the TS/MP 2.0 LINKMON process. For more information on ACS subsystem limits, see the *TS/MP 2.7 ACS Reference Manual*.

Extended Memory

The operating system manages the extended memory requirements of the ACS subsystem. Therefore, an external swap file is not required.

ACS Subsystem Memory Dumps

Unlike TS/MP 2.0, the ACS subsystem core processes no longer halt the processor in case of internal product errors. Instead, ACS core processes stop themselves after creating memory abend files.

NOTE: If one of the ACS core processes on a processor goes down, the remaining core processes on the same processor also go down. Issuing the `SCF CONTROL` command can restart the stopped core process on the processor. For more information about the `SCF CONTROL` command, see the *TS/MP 2.7 ACS Reference Manual*.

Pathsend Management Tasks

To communication between Pathsend processes and server processes, you must configure your PATHMON environment to specify:

- The number of ACS subsystem processes that your PATHMON environment can support at one time
- The appropriate security to control access to server classes

You cannot configure or control ACS subsystem processes with PATHCOM commands in the same way that you control and configure PATHMON-controlled objects. However, you can, monitor the ACS subsystem processes, as described later in this section.

Specifying the Maximum Number of Link Manager (LINKMON/ACS Subsystem) Processes

Before starting a PATHMON environment, you specify the maximum number of link manager (LINKMON/ACS subsystem) processes that the PATHMON environment can support at one time by setting the *MAXLINKMONS* parameter with the `SET PATHWAY` command.

This example specifies that a maximum of 16 link manager (LINKMON/ACS subsystem) processes can access your PATHMON environment:

```
= SET PATHWAY MAXLINKMONS 16
```

If you have Pathsend processes in other systems that need to access server classes in your PATHMON environment, setting the MAXLINKMONS attribute to the number of CPUs in your system might not be correct.

NOTE:

If the MAXLINKMONS attribute is set to 0, any attempt by a Pathsend process to perform a server-class send operation to a server class in the PATHMON environment fails. The default value of the MAXLINKMONS attribute is 0.

Specifying Security

For PATHMON environments that use Pathsend, there are two levels of security to consider:

- Network-level security
- Server-class-level security

Network Level Security

For Pathsend processes to access server processes, this network conditions must be met:

- The link manager (LINKMON/ACS subsystem) process must be able to open the PATHMON process (to make link requests).
- The link manager (LINKMON/ACS subsystem) process must be able to open the server processes (to send user requests).
- The PATHMON process must be able to open the server processes (to send startup messages).

All of these opens are performed using the PATHMON process's user ID. Therefore, the user ID of the PATHMON process controlling the server class must have corresponding user IDs on—and remote passwords with—these systems:

- The system of the Pathsend process
- The PATHMON process's system
- The server class's system

Server Class Security

To configure servers for access by Pathsend processes, you use the `SET SERVER` command to define attributes for the OWNER and SECURITY parameters. Together, the OWNER and SECURITY parameters define whether a Pathsend process can access a server class.

Link manager (LINKMON/ACS subsystem) processes perform authorization checks on each send to make sure that the user ID of the Pathsend process at the time of the send conforms to the SERVER OWNER and SERVER SECURITY attributes.

(Note that TCPs running under the Pathway/iTS product ignore these parameters; the parameters only affect Pathsend process access.)

OWNER

The *SERVER OWNER* parameter defines the user ID that controls access to the server class from a Pathsend process.

The user ID you assign must be known to the system in which PATHCOM is running. You define the user ID either with a system number, group number, and user number, or with a system name, group name, and user name.

For example, these commands set the SERVER OWNER attribute to the user ID of the super user on the system 222 or \ABC, respectively:

```
= SET SERVER OWNER 222.255,255
= SET SERVER OWNER \ABC.SUPER.SUPER
```

If the user ID's system is the same as the system in which PATHCOM is running, you need not specify the system number or name. If you do not specify the *SERVER OWNER* parameter, the default user ID is the owner ID of the user who started the PATHMON process.

SECURITY

The SERVER SECURITY attribute specifies the users, relative to the SERVER OWNER, who can access a server class from a Pathsend process.

You set the *SERVER SECURITY* parameter using the Guardian security values A, G, O, -, N, C, and U.

For example, this command allows only those Pathsend processes running under the process ID and on the same system as the user specified by the SERVER OWNER attribute to access the server class:

```
= SET SERVER SECURITY "O"
```

"N" is the default security for the SERVER SECURITY attribute. Any Pathsend process with the appropriate network security (described earlier in this section) can access the server class.

Altering Server Class Security

Once your PATHMON environment is configured and running, you can use the `ALTER SERVER` command to change the values of SERVER OWNER and SERVER SECURITY. For example,

```
= ALTER SERVER CLASS-1, OWNER 123,45, SECURITY "O"
```

A server class must be stopped before you issue the `ALTER` command.

Monitoring Link Manager (LINKMON/ACS subsystem) Processes

Using PATHCOM, you cannot control or configure LINKMON processes. You can, however:

- Check the value for the *MAXLINKMONS* parameter.
- Display information about link manager (LINKMON/ACS subsystem) processes communicating with your PATHMON environment.
- Display statistics for server classes to which link manager (LINKMON/ACS subsystem) processes have links.

Checking the Value for MAXLINKMONS

To check the value that you defined for MAXLINKMONS and to check the number of link manager (LINKMON/ACS subsystem) processes currently accessing your PATHMON environment, use the `INFO PATHWAY` command, as shown in this example:

```
= INFO PATHWAY
```

If you set MAXLINKMONS equal to 16 and there are currently four link manager (LINKMON/ACS subsystem) processes that have one or more links to server classes within your PATHMON environment, the `INFO PATHWAY` command returns this information for the *MAXLINKMONS* parameter:

```
MAXLINKMONS 16 [ CURRENTLY 4]
```

Displaying Information About ACS Subsystem Processes

You can also display information about ACS subsystem processes using these STATUS commands:

- STATUS LINKMON
- STATUS PATHMON
- STATUS PATHWAY
- STATUS SERVER

(For complete examples of display output for the `STATUS PATHWAY`, `STATUS PATHMON`, and `STATUS SERVER` commands, see **Maintaining a PATHMON Environment** on page 81.)

STATUS LINKMON Command

To get status information about the LINKMON processes communicating with your PATHMON environment, use the `STATUS LINKMON` command.

This command displays status information about a particular LINKMON process running on a specific system:

```
= STATUS LINKMON L\ABC.$ZL05
```

The next command directs status information to a file:

```
= STATUS/OUT STATFLE/LINKMON L\CPTNO.$ZL05
```

The next command displays status information, as shown in **STATUS LINKMON Display** on page 117, for all LINKMON processes communicating with the PATHMON process:

```
= STATUS LINKMON *
```

STATUS LINKMON Display

LINKMON	STATE	ERROR INFO	PROCESS	CPU
L\SYS.$ZL01	RUNNING		$ZL01	1
L\SYS.$ZL04	RUNNING		$ZL04	4
L\SYS.$ZL05	RUNNING		$ZL05	5
L\SYS.$ZL06	RUNNING		$ZL06	6
L\SYS.$ZL08	RUNNING		$ZL08	8

Table Continued

L\SYS.$ZL09	RUNNING	$ZL09	9
L\SYS.$ZL10	RUNNING	$ZL10	10
L\SYS.$ZL11	RUNNING	$ZL11	11

STATUS PATHMON Command

To identify the link manager (LINKMON/ACS subsystem) processes currently communicating with your PATHMON environment, use the `STATUS PATHMON` command:

```
= STATUS PATHMON
```

The STATUS PATHMON display shows the Guardian process ID of the link manager processes and their process-accessor IDs, as shown in **LINKMON Information in STATUS PATHMON Display** on page 118.

LINKMON Information in STATUS PATHMON Display

```
PATHMON \CHI.$PM32 -- STATE=RUNNING          CPUS 8:0
  PATHCTL   (OPEN)     $OPER.TRANCNFG.PATHCTL
  LOG1      (CLOSED)   $0
  LOG2      (CLOSED)
  REQNUM FILE       PID             PAID     WAIT
                .
                .
                .
  15     LINKMON    $ZL11           30,1
                .
                .
                .
```

STATUS PATHWAY Command

To determine the number of LINKMON processes communicating with your PATHMON environment, use the `STATUS PATHWAY` command:

```
= STATUS PATHWAY
```

LINKMON Information in STATUS PATHWAY Display on page 118 shows that there is 1 LINKMON process communicating with this PATHMON environment.

LINKMON Information in STATUS PATHWAY Display

```
PATHCOM results:
PATHWAY -- STATE=RUNNING
                 RUNNING
              .
              .
              .
LINKMONS              1
              .
              .
              .
```

STATUS SERVER Command

To identify the link manager (LINKMON/ACS subsystem) processes currently linked to a specific server class, use the `STATUS SERVER` command with the `PROCESSES` option:

```
= STATUS SERVER PROCESS-SERVER, PROCESSES
```

The preceding command displays the status of all server processes within server class CLASS-1, including the names of the LINKMON/ACS processes currently linked to the server, as shown in **LINKMON Information in STATUS SERVER Display** on page 119.

LINKMON Information in STATUS SERVER Display

```
SERVER            #RUNNING ERROR INFO
PROCESS-SERVER        4    3115  34
                      .
                      .
                      .
  PROCESS           STATE    ERROR INFO #LINKS WEIGHT
  $ZTP2             RUNNING                 4     9

     LINKER           LINK COUNT
     M6530-TCP1       003
     L\SYS.$ZL11      001
                       .
                       .
                       .
```

STATS SERVER Command

To display statistics collected by ACS subsystem processes, use the `STATS SERVER` command.

ACS subsystem processes are continually gathering statistics about each server class with which they have links, unless the server class stops for any reason—in which case, the statistics are lost.

The statistics displayed when you issue a `STATS SERVER` command show the ACS subsystem process resources used to communicate with the server class, including information about queues and sends to and replies from the server class.

For more information about server statistics, see **Tuning Your System by Using Statistics** on page 120.

Tuning Your System by Using Statistics

This section describes statistics collected for server processes. These statistics are collected when you run the `STATS SERVER` command. See the description of the **STATS SERVER Command** on page 233.

This section is divided into two parts:

- The first part describes server process performance statistics that the ACS subsystem processes, and the TCP collect.
- The second part describes server process performance statistics collected only by the TCPs. Because TCPs are provided by the Pathway/iTS product, you can display these server statistics only if your environment includes the Pathway/iTS product.

You can use the statistics generated by transaction processing functions in the PATHMON environment to detect bottlenecks within the ACS subsystem processes or—if you have Pathway/iTS—the TCP. Based on careful analysis of these statistics, you can reconfigure your PATHMON environment to eliminate some performance degradations.

Other factors affecting system performance are not represented in these statistics. Such factors generally involve contention for system resources, for example, the ability of the PATHMON process to grant links promptly.

Statistics produced by the NonStop operating system can shed light on the performance of these types of functions. For more information about statistics produced by the operating system, see the *Measure User's Guide and the Measure Reference Manual*.

Server Statistics Collected by the ACS Subsystem Processes and the TCP

ACS subsystem processes and TCPs gather statistics for a single server class or for multiple server classes. The statistics are gathered by:

- ACS subsystem processes with links to a server process in the class
- TCPs that open a server process in the class—but only if these TCPs are configured with the

  STATS

  parameter set to ON

Server statistics show the amount of link manager resources used to communicate with a server class. Server statistics collected by ACS subsystem processes and TCPs provide information about:

- Server-class wait queue (QUEUE INFO)
- Input and output operations (I/OþINFO)

Sample Server Statistics on page 121 shows a sample set of server statistics gathered by a TCP (TCP1) and a ACS subsystem process (LM1).

Sample Server Statistics

```
SERVER                                                  19 FEB       15:22:30
SERVME IN                                               1996,
TCP TCP1
*



QUEUE        REQ CNT      % WAIT       MAX WAITS    AVG WAITS    % DYNAMIC
INFO*:



             33272        99.8         7            2.37         99.9

I/O INFO*:   REQ CNT      MAX TSIZE    AVG TSIZE    I/O CNT


SEND         33304        2            2            33304

REPLY                     2            2

RESPONSE TIME INFO (TIME VALUES IN SECS)*:


SEND TO SERVERCLASS

SUMMARY      # MEAS       AVG RESP     MAX RESP     MIN RESP     STAND DEV

             33303        0.50         2.51         0.10         0.49

IN ACS subsystem LM1 INTERVAL 10 MINS COUNT 1/5 25 SEP 1984, 11:50:25

QUEUE INFO:  REQ CNT      % WAIT       MAX WAITS    AVG WAITS    % DYNAMIC

             19           0.0          0            0.00          94.7

I/O INFO:    REQ CNT      MAX TSIZE    AVG TSIZE    I/O CNT

SEND         2602         1409         155          2602

REPLY                     1877         455

* Collected only if you have the Pathway/iTS product and TCP STATS ON
```

Note that neither the ACS subsystem process nor the TCP generate values for the REPLY REQ CNT and REPLY I/O CNT fields. Note also the inclusion of Response Time Info. These statistics are collected only the TCP. See **Response Time Info** on page 125.

Queue Info

The QUEUE INFO section in the statistics display provides information about the server-class wait queue.

This queue is for SEND operations that the link manager such as ACS subsystem processes, or TCP, cannot run immediately because a link to the server class is not available. Server class requests wait in the link manager until an existing link becomes available or until the PATHMON process grants a new link.

The following table shows only the QUEUE INFO section of the sample server statistics. See this figure as you read this section.

QUEUE INFO Section of Sample Server Statistics

```
SERVER
SERVME

IN TCP TCP1                                          19 FEB       15:22:30
*                                                    1996,

QUEUE INFO    REQ CNT      % WAIT       MAX WAITS    AVG WAITS    % DYNAMIC
*

:

              33272        99.8         7            2.37         99.9

IN ACS subsystem LM1 INTERVAL 10 MINS COUNT 1/5 25 SEP 1984, 11:50:25

QUEUE INFO:   REQ CNT      % WAIT       MAX WAITS    AVG WAITS    % DYNAMIC

              19           0.0          0            0.00         94.7
```

* Collected only if you have the Pathway/iTS product and TCP STATS ON

The following table shows the statistics for the server-class wait queue and what they indicate. If these statistics show that requests are queuing, you might consider steps to reduce queuing and improve response time. Such steps might include reconfiguring your system or modifying the application.

Table 6: Server Statistics for QUEUE INFO

Statistic	For Server-Class Wait Queue, Indicates:
REQ CNT	Number of requests for a link that could not be satisfied immediately because of the unavailability of a link or when the link manager is in the process of requesting a link from the process broker (PB process) in the same system.

% WAIT	Percentage of requests for a link (based on REQ CNT) that the ACS subsystem processes queue when one or more other requests are already queued for the server class.
	For example:
	`A TCP sends 100 requests to a serverclass. It had to queue 10 of them because no link was immediately available.`
	`For two of the 10 queued requests, there was already at least one entry in the queue when the new entry was to be queued. In this case, the Queue Info %Wait will be 20%(2/10), not 10% (10/100) or 2% (2/100)."`

MAX WAITS	The maximum number of link requests that were queued in a link manager at any point.
AVG WAITS	How many requests were waiting, on average, during the sample period.
% DYNAMIC	Percentage of requests for a link (based on REQ CNT) that the ACS subsystem processes received and resulted into a dynamic link.

REQ CNT

REQ CNT indicates the number of requests for a link to a server that could not be satisfied immediately because of the unavailability of a link or when the link manager is in the process of requesting a link from the process broker (PB process) in the same system.

You can compare this number with the REQ CNT number for I/O INFO (see **I/O Info** on page 123 earlier in this section) to determine what percentage of the transactions requesting a link to a particular server experienced some delay.

% DYNAMIC

The higher the number indicated for % DYNAMIC, then the greater the chance that a dynamic server will be required. It is important to realize that this number does not indicate:

- That dynamic servers were in fact started. However, it does indicate, that some requests were delayed anywhere from not at all to the time specified with the `SET SERVER CREATEDELAY` command.
- The number of dynamic links granted by the PATHMON process. Any request on the LINK queue can be cancelled by the availability of another existing link.

If the number for %DYNAMIC is large, you might want to increase the number of static servers.

I/O Info

The I/O INFO section in the statistics display provides information about I/O operations to and from the server class.

I/O INFO Section of Sample Server Statistics on page 124 shows only the I/O INFO section of the sample server statistics. See this figure as you read this section.

I/O INFO Section of Sample Server Statistics

```
SERVER
SERVME

IN TCP TCP1                                          19 FEB 1996, 15:22:30
*

I/O INFO     REQ CNT      MAX TSIZE    AVG TSIZE    I/O CNT
*

:

SEND         33304        2            2            33304

REPLY        ..           2            2

                                       .

                                       .

IN ACS subsystem LM1 INTERVAL 10 MINS COUNT 1/5 25 SEP 1984, 11:50:25

             REQ CNT      MAX TSIZE    AVG TSIZE    I/O CNT
I/O INFO:

SEND                      1409         155          2602
             2602

REPLY                                  455
                          1877

* Collected only if you have the Pathway/iTS product and TCP STATS ON
```

The following table shows the server statistics for I/O operations and what they indicate.

Table 7: Server Statistics for I/O Info

Statistic	For SEND, Indicates:	For REPLY, Indicates:
REQ CNT	Total number of Pathsends to servers in a server class. Pathsends include any of the following APIs: SERVERCLASS_DIALOG_ BEGIN_, SERVERCLASS_DIALOG_ SEND_, SERVERCLASS_SEND_, and SERVERCLASS_SENDL_ .	N.A.
MAX TSIZE	Size, in bytes, of the largest amount of data transferred in a send.	Size, in bytes, of the largest amount of data transferred in a reply from the server class.
AVG TSIZE	Size, in bytes, of the average amount of data transferred during a send.	Size, in bytes, of the average amount of data transferred in a reply from the server class.
I/O CNT	Number of I/O operations to servers of this server class. (For `SEND` commands, I/O CNT is always the same as the REQ CNT.)	N.A

REQ CNT

Statistics are not provided for REPLY statements because there are no explicit requests for REPLY.

I/O CNT

Statistics are not provided for REPLY statements because there are no explicit requests for REPLY.

Server Statistics Collected Only By the TCP

Server statistics show the amount of link manager resources used to communicate with a server class. Only TCPs collect server statistics that provide information about:

- Response time (RESPONSE TIME INFO)
- Frequency distribution (optional)

These statistics are collected only if your environment includes the Pathway/iTS product.

Response Time Info

The RESPONSE TIME INFO section of the statistics display provides information about the completion of SCREEN COBOL SEND statements.

The measurements are gathered by the TCP for the servers with which it has links. As operations proceed, the TCP captures each measurement by starting an internal clock the moment after it issues a SEND statement and stopping the clock the moment after it receives a reply from the server process. The captured measurements represent server response time.

RESPONSE TIME INFO Section of Sample Server Statistics on page 126 shows only the RESPONSE TIME INFO section of the sample server statistics. See this example as you read this section.

RESPONSE TIME INFO Section of Sample Server Statistics

```
SERVER SERVME?

IN TCP TCP1                                          19 FEB 1996, 15:22:30
*

                                        .

                                        .

                                        .

RESPONSE TIME INFO (TIME VALUES IN SECS)
*

:

SEND TO SERVERCLASS

SUMMARY #    MEAS          AVG RESP      MAX RESP      MIN RESP      STAND DEV

             33303         0.50          2.51          0.10          0.49

                                        .

                                        .

                                        .
```

* Collected only if you have the Pathway/iTS product and TCP STATS ON

The following table shows the response time statistics and what they indicate.

Table 8: Server 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

If the number for STAND DEV is large in comparison to the number for AVG RESP, then you must review your application to determine the cause.

These limits are imposed on measurement collection:

- Measurements will stop for any server upon the 99,999,999th measurement.
- Any measurement greater than 99,999,999.99 seconds is discarded.
- After a measurement counter reaches 99,999,999.99 seconds, measurements will stop being collected for this server.
- Response times are rounded to the nearest hundredth of a second.

Frequency Distribution

When you specify the `FREQTABLE` option of the `STATS SERVER` command, the PATHMON process displays supplemental statistics on the completion of SCREEN COBOL SEND statements within a given time interval. These statistics are collected only if your environment includes the Pathway/iTS product.

The supplemental information is contained in the frequency distribution table, as shown in **Sample Server Statistics With Frequency Distribution Table** on page 127. A separate frequency distribution table is generated for each server class.

Sample Server Statistics With Frequency Distribution Table

```
SERVER
SERVME

IN TCP TCP1*                19 FEB       15:26:14
                            1996,


QUEUE INFO    REQ CNT       % WAIT       MAX WAITS     AVG WAITS     % DYNAMIC
*

:

                            99.8         7             2.38          9

I/O INFO      REQ CNT       MAX TSIZE    AVG TSIZE     I/O CNT
*

:

SEND          33740         2            2             33740

                            2            2

RESPONSE TIME INFO (TIME VALUES IN SECS)
*

:

SEND TO SERVERCLASS

SUMMARY                     AVG RESP     MAX RESP      MIN RESP      STAND DEV
               # MEAS

               33739        0.50         2.51          0.01          0.49
```

Table Continued

```
FREQUENCY DISTRIBUTION
*
:
TIME INTERVAL (0.05 SECS                    # MEAS       CUM %
TI01 <0.01                                  132          0.3
0.01 <= TI02 <0.05                          16720        49.9
0.05 <= TI03 < 0.11                         5            49.9
0.11 <= TI04 < 0.17                         1            49.9
0.17 <= TI05 < 0.22                         9            49.9
0.22 <= TI06 < 0.28                         0            49.9
0.28 <= TI07 < 0.34                         0            49.9
0.34 <= TI08 < 0.39                         1            49.9
0.39 <= TI09 < 0.45                         1            49.9
0.45 <= TI10 < 0.51                         0            49.9
0.51 <= TI11 < 0.56                         0            49.9
0.56 <= TI12 < 0.62                         0            49.9
0.62 <= TI13 < 0.68                         0            49.9
0.68 <= TI14 < 0.73                         0            49.9
0.73 <= TI15 < 0.79                         0            49.9
0.79 <= TI16 < 0.85                         0            49.9
0.85 <= TI17 < 0.90                         0            49.9
0.90 <= TI18 < 0.96                         0            49.9
0.96 <= TI19 < 1.02                         16472        98.8
1.02 <= TI20                                398          100.0
*
```

Collected only if you have the Pathway/iTS product and TCP STATS ON

The following table shows the server statistics for frequency distribution and what they indicate.

Table 9: Server 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

You can use the statistics for frequency distribution to determine the times of peak load for your system. The example in **Sample Server Statistics With Frequency Distribution Table** on page 127 shows a bimodal distribution with one peak almost 20 times as long as the other. This is an indication that some system level tuning might be required.

The frequency distribution table is not generated if either of these are true:

- There are fewer than 50 sample measurements collected at the time of the STATS request. In this case, this text is displayed at the bottom of the STATS SERVER 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, this is displayed at the bottom of the STATS SERVER display:

  ```
  FREQUENCY DISTRIBUTION:? NOT PRODUCED: COMPUTED INTERVAL
  LESS THAN .01 SECS
  ```

Overview of PATHCOM

This section contains overview information to help you get started using PATHCOM to manage your PATHMON environment. For more detailed information on how PATHCOM works and how to manage your Pathway environment, see **Introduction to NonStop TS/MP System Management** on page 14 through **Tuning Your System by Using Statistics** on page 120.

NOTE:

- This manual provides information specific to commands dealing with objects managed through the NonStop TS/MP product. PATHCOM commands dealing with SCREEN COBOL requestors, such as those operating on TCP, TERM, and PROGRAM objects, which are controlled though the Pathway/iTS product, are described in *Pathway/iTS System Management Manual*.
- When using TS/MP 2.6 or later, Hewlett Packard Enterprise recommends that you use PDMCOM instead of PATHCOM, because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference 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 server class is not configured, the `START` command for that object has no effect because that object cannot be started until it is configured using the `SET` and `ADD` commands. **Figure 22: PATHCOM Commands and Object States** on page 130 illustrates the relationship between the state of an object and the function of the PATHCOM commands.

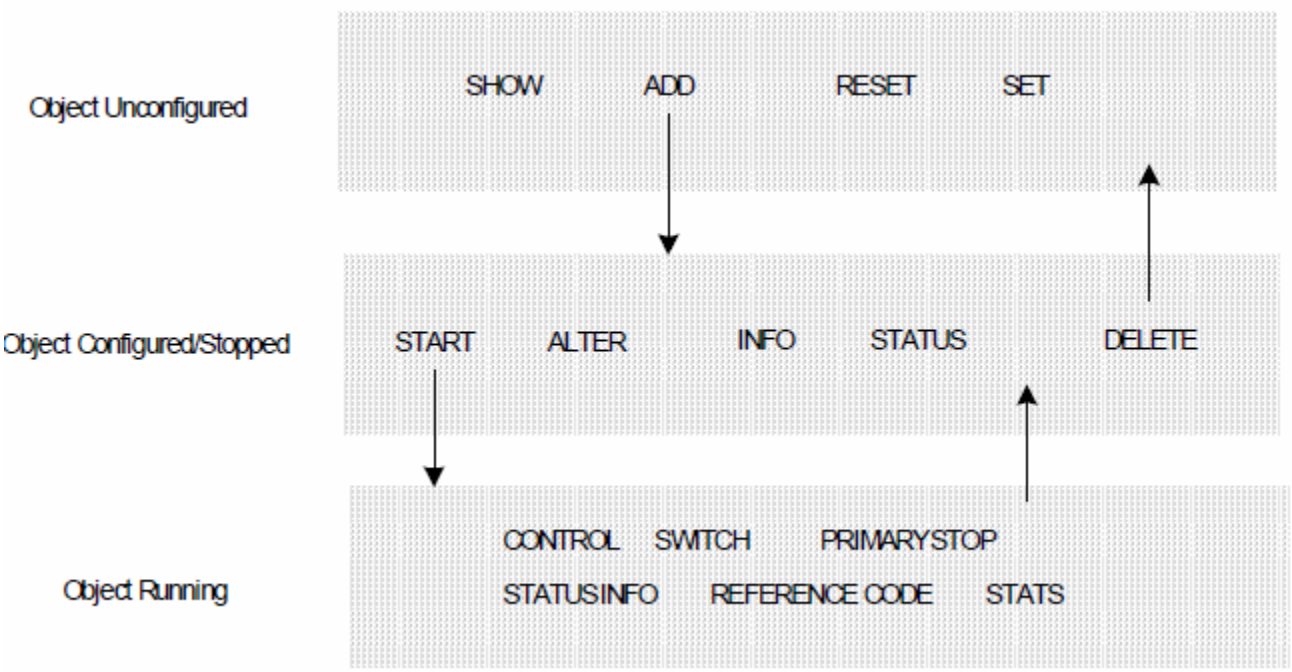

Figure 22: PATHCOM Commands and Object States

NOTE:

The `CONTROL TERMs` command must be issued when TERM is in the STOPPED state for COUPLE request.

The relationships illustrated in **Figure 22: PATHCOM Commands and Object States** on page 130 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 `CONTROL` command affects only the PATHMON process and the `FREEZE SERVER` command affects only server classes.

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

PATHCOM Commands is a list of the commands you can run using PATHCOM. The table entries provide the name of the command, its description, and the names of the objects influenced by the command.

NOTE: The commands in this table pertain only to objects managed through the NonStop TS/MP product. For commands for objects managed under the Pathway/iTS product, see *Pathway/iTS System Management Manual*.

Table 10: PATHCOM Commands

PATHCOM Commands	Description	Valid Objects
`ADD`	Enters a description of an object into the configuration	SERVER
`ALTER`	Changes the attributes of a previously defined object while the object is stopped	SERVER
`CMDCWD`	Sets the default absolute OSS directory for expansion of any OSS process file names.	
`CMDVOL`	Sets the default volume and subvolume for expansion of any Guardian process file names except Command file names	
`CONTROL*`	Changes the attributes of an object while it is running	PATHMON process
`DELETE`	Removes an object description from the system configuration	SERVER

Table Continued

PATHCOM Commands	Description	Valid Objects
ERRORS	s PATHCOM to continue executing commands after encountering a specified number of errors	
EXIT	Stops PATHCOM	
FC	Provides the ability to edit or to repeat a command-line	
FREEZE	Prohibits communication between a terminal and a server class	SERVER
HELP	Displays or writes the syntax of the PATHCOM commands and the text of Pathway error messages to a file	
INFO	Displays values for object attributes	PATHMON process, PATHWAY, SERVER
LOG1 and LOG2	Specifies the files used for reporting errors and changes in status	PATHMON process
OBEY	Causes commands to be read from a specified file	
OBEYVOL	Sets the default volume and subvolume for expansion of command file names	
OPEN	Specifies the name of the PATHMON process to which PATHCOM commands are directed	
PRIMARY	Causes primary and backup processes to resume executing in the CPUs defined in the object configuration	PATHMON process
RESET	Resets object attributes to their default values	CMDCWD, SERVER
SET	Establishes values for object attributes prior to adding an object to the Pathway environment	PATHMON process, PATHWAY, SERVER
SHOW	Displays current values for the object attributes and environmental settings for PATHCOM	CMDCWD, CMDVOL, SERVER
SHUTDOWN	Stops all PATHMON-controlled objects except PATHCOM	PATHMON process

Table Continued

PATHCOM Commands	Description	Valid Objects
SHUTDOWN2	Stops all PATHMON-controlled objects except PATHCOM	PATHMON process
START	Starts one or more objects	PATHWAY process, SERVER
STATS	Displays resource usage and system performance statistics	SERVER
STATUS	Displays the current status of any PATHMON-controlled object or a LINKMON process linked to a server class	LINKMON*, PATHMON process, PATHWAY, SERVER
	* Although LINKMON is listed here as a valid object, it is not like other NonStop TS/MP objects. LINKMON is not a PATHCOM object because it cannot be controlled by PATHCOM commands; PATHCOM can only obtain status information about LINKMON.	
STOP	Stops a PATHMON-controlled object	PATHMON process, SERVER
SWITCH	Causes primary and backup processes to exchange functions	PATHMON process
THAW	Removes the prohibition imposed with the FREEZE command on use of a server class	SERVER
*The CONTROL TERMs command must be issued when TERM is in the STOPPED state for COUPLE request.		

Command and Object Relationships

The following table summarizes the relationship between the PATHCOM commands and the different PATHMON-controlled objects, the PATHCOM process, ACS subsystem processes, and tell messages.

Table 11: Commands and Objects

Objects PATHCOM Commands	PATHWAY	PATHMON	SERVER	LINKMON
ADD			x	
ALTER			x	
CONTROL		x		
DELETE			x	

Table Continued

Objects PATHCOM Commands	PATHWAY	PATHMON	SERVER	LINKMON
FREEZE			x	
INFO	x	x	x	
LOG1, LOG2		x		
PRIMARY		x		
RESET			x	
SET	x	x	x	
SHOW			x	
SHUTDOWN		x		
SHUTDOWN2		x		
START	x		x	
STATS			x	
STATUS	x	x	x	x
STOP		x	x	
SWITCH		x		
THAW			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, this command:

  ```
  SET SERVER PROCESSTYPE GUARDIAN, AUTORESTART 1, PROGRAM
  $MKT.PR6
  ```

  can also be entered as:

  ```
  SET SERVER PROGRAM $MKT.PR6, PROCESSTYPE GUARDIAN,
  AUTORESTART 1
  ```

  Exceptions to the independent positioning of options within a command are documented with the individual commands.

- The SERVER attribute keywords and values for ARGLIST, ASSIGN, DEFINE, ENV, and PARAM must be enclosed in parentheses when multiple arguments (values) are specified for the attribute and other

attributes are included in the same command. Parentheses are not required if the attribute is the last specified in the command.

These examples show correct usage of parentheses. In the first command parentheses are not required because only one attribute is defined. In the second example, the first attribute is enclosed in parentheses because another attribute is specified after it; the ASSIGN definition is the last in the list and does not require parentheses. In the third example, both attributes must be enclosed in parentheses because neither is the last in the list.

```
SET SERVER PARAM A a, B b
SET SERVER ( PARAM A a, B b ), ASSIGN aaaaa, bbbbb
SET SERVER ( PARAM A a, B b ), ( ASSIGN aaaaa, bbbbb ) , &
PRI 141
```

- 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.
  =ADD SERVER B-SERVE [Here is a comment.],LIKE A-SERVE
  ```

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. For information on including comments in TACL scripts, see the `TACL Reference Manual`.

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 runs the command and issues another prompt.

In this example, the first command starts the PATHMON process, the second command starts the PATHCOM process, the third line shows the PATHCOM sign on, and the fourth line shows the PATHCOM prompt:

```
TACL 8> PATHMON/NAME $PM1,CPU 0,NOWAIT/
TACL 9> PATHCOM $PM1
PATHCOM - T9153C31 - (08SEP92)
=
```

You can group two or more commands separated by semicolons (;). For example, these two commands can be entered in two ways, as follows:

```
= command-1; command-2
```

or:

```
= command-1
= command-2
```

Pressing the terminal Break key during the execution of a PATHCOM command (described in **PATHCOM Operation Commands** on page 144) 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, dynamically start and stop PATHMON-controlled objects, and inform the Pathway terminal operators about changing conditions using tell messages.

Noninteractive Mode

PATHCOM functions in noninteractive mode when it reads commands from a command file.

In this example, PATHCOM reads commands from a file named CMDFILE and lists them on the device $S.#LP. When it encounters an end-of-file command or an EXIT command, PATHCOM terminates.

```
3> PATHCOM/IN CMDFILE,OUT $S.#LP, CPU 1, NOWAIT/$PM2
```

You can use DEFINEs to specify names for the files that PATHCOM uses directly as IN, OUT, and OBEY command files. In this 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, see the *TACL Reference Manual* and **the** *Guardian User’s Guide*.

Guardian File Names

Each disk file in the Guardian file-system is identified by a unique, symbolic file name, described in these 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
- **sub** volume—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 Hewlett Packard Enterprise subsystem programs 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 Pathway environment is running on this system, only $TAPE1 is required for the file name.

File Name Expansion

File name expansion is accomplished through the use of default names. If you omit the node, volume, or subvolume name in a command parameter, PATHCOM uses system defaults to expand the file name before passing the name to the PATHMON process.

To get the default for the node name, PATHCOM first determines whether the SET PATHWAY NODEINDEPENDENT attribute is set to ON. If NODEINDEPENDENT is ON, the default node name is * , a generic name representing the node where the PATHMON process is currently running. For information on configuring for the node independence, see **Specifying Node Independence** on page 53 and **SET PATHWAY Command** on page 171.

If the SET PATHWAY NODEINDEPENDENT attribute is set to OFF, PATHCOM uses the node name you supply in the `CMDVOL` command. PATHCOM also uses the default volume and subvolume names specified in the `CMDVOL` command. For information on the `CMDVOL` command, see **CMDVOL Command** on page 145.

Although the PATHMON process uses system names, it does not store server process names (nor Pathway/iTS TCP names) in fully qualified network form. All other file names are kept in the form:

`\node.$volume.subvolume.file-identifier`

OSS Pathnames

An OSS pathname in the OSS attributes PROGRAM, STDIN, STDERR, and STDOUT can be an absolute pathname or a relative pathname. Case is significant.

When specifying an OSS pathname longer than one line, use the PATHCOM continuation character (&) at the end of the line to indicate continuous input. If the pathname includes an ampersand (&) as part of the name and the ampersand occurs at the end of an input line, enter two ampersands. An ampersand occurring in the middle of an input line, embedded in a pathname, need not be doubled. To embed blanks, quotes, commas, and semicolons, use the same rules as for PATHCOM, and enclose the entire pathname in quotes.

Absolute Pathnames

Absolute pathnames must begin with a forward slash (/) and can have a maximum length of 1023 characters; the last character (character 1024) is reserved for a null.

When setting an OSS pathname using the `SET SERVER CWD` command or when setting a default directory using the `CMDCWD` command, you must specify an absolute pathname.

Absolute pathnames specified for server process attributes are validated when the `SET SERVER` command is processed.

Relative Pathnames

At process time, PATHCOM treats any OSS pathname that does not begin with a forward slash (/) as a relative pathname and expands it to create an absolute pathname.

When expanding a relative pathname, PATHCOM adds a forward slash and attaches the OSS directory specified in the `SET SERVER CWD` command. If no directory is set, the default set with the `CMDCWD` command is used. If the value of the CWD attribute ends with a forward slash, then PATHCOM does not add another one.

Relative pathnames specified for server process attributes are initially validated when the `SET SERVER` command is processed; additional validation is performed during the execution of the `ADD SERVER` and `ALTER SERVER` commands. An error message is generated at the time an `ADD SERVER` or `ALTER SERVER` command is processed if either of these conditions apply:

- The expansion of a relative pathname results in an absolute pathname greater than the maximum 1023 characters allowed.
- An expanded pathname has a forward slash as the final character.

CAUTION:

NonStop TS/MP makes no changes to existing OSS path names. If escape sequences or other nonprintable characters are included in a pathname, unexpected results might occur. Hewlett Packard Enterprise recommends that all characters in OSS path names be ASCII characters. One exception: to use a dollar sign ($) in an OSS pathname, you must use an escape character because the OSS shell treats $ as a special character.

PATHMON and PATHCOM Startup Commands

This section describes the TACL commands for starting the PATHMON process and PATHCOM.

Before starting PATHCOM, you must start the PATHMON process, the central control process of a Pathway environment. The default name of the PATHMON object file is $SYSTEM.SYSTEM.PATHMON.

NOTE: When using TS/MP 2.6 or later, Hewlett Packard Enterprise recommends that you use PDMCOM instead of PATHCOM because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

Starting the PATHMON Process

You start the PATHMON process by entering the PATHMON command through the TACL command interpreter.

```
PATHMON / NAME $process-name [ , run-option ]... /
```

```
run-option is:


     CPU number
     INSPECT ON | SAVEABEND
     NOWAIT
     OUT log1-file
     PRI number
```

$process-name

is the name of the PATHMON process. The name can be a maximum of 6 characters. $

process-name

begins with a dollar sign ($), followed by 1 to 5 alphanumeric characters. The first alphanumeric character must be a letter.

If the PATHMON process name is to be used across a network, $process-name can be no longer than 5 characters, that is, a dollar sign ($) followed by no more than 4 alphanumeric characters (for example, $abc2).

This parameter is required.

NOTE: If multiple Pathway environments are configured on a node, do not specify the name $PM for any PATHMON process on that node. Because $PM is the default name used when someone opens communication with PATHCOM without specifying a PATHMON name, your commands might be applied to the wrong Pathway environment.

run-option

is one of the TACL RUN [D] command run options. For details about the RUN [D] run options, see the *TACL Reference Manual*.

CPU number

specifies the processor where the primary PATHMON process runs. This number can be the number of any processor on your system from 0 through 15. The backup processor is specified with the `SET PATHMON` command and the *BACKUPCPU* parameter.

If you omit this parameter, the default is the processor used by the TACL command interpreter. Using the default parameter can cause an error later if the PATHMON process attempts to start its backup process in the same processor.

`INSPECT ON | SAVEABEND`

establishes default debugging conditions for the process. Hewlett Packard Enterprise recommends the following values:

`ON`

directs the file system to use Inspect if a debug operation is required.

`SAVEABEND`

directs Inspect to create a copy of a process if it abends, and save it in a separate file for debugging purposes.

If you start the PATHMON process with INSPECT ON and configure a server class with the `SET SERVER DEBUG ON` option, the server process within the class enters debug mode at the time the server process begins operation.

`NOWAIT`

specifies that the TACL command interpreter is not to wait for the PATHMON process to terminate, but is to return immediately with a command input prompt. Hewlett Packard Enterprise recommends this option, because the PATHMON process does not terminate until all objects controlled by the PATHMON process have shut down.

`OUT` **`log1-file`**

specifies the name of the LOG1 file that is used for logging and reporting errors and changes in status. If you specify a disk file, use the TACL command interpreter or the `FUP CREATE` command to create the file before starting the PATHMON process. EDIT files are not acceptable log files to the PATHMON process.

This parameter is equivalent to the `LOG1` command described in **PATHMON Environment Control Commands** on page 159.

If you omit this parameter, output is directed to the TACL list file; this is typically the home terminal. In a production environment, Hewlett Packard Enterprise recommends that you specify a log file.

`PRI` **`number`**

specifies the execution priority of the PATHMON process.

`number`

can be a value from 1 through 199. If you omit this parameter, the default is one less than the priority of the TACL command interpreter. If there is an active $CMON on the system, the default is the priority $CMON assigns.

Examples

These commands start the PATHMON process using various parameters:

```
PATHMON /NAME $POM, NOWAIT, CPU 3/
PATHMON /NAME $PAT, NOWAIT/
PATHMON /NAME $PUT, NOWAIT, CPU 7, INSPECT ON, OUT $TERM1/
```

Starting PATHCOM

PATHCOM is the process that provides the interactive command interface to the PATHMON process. You start a PATHCOM process after first starting the PATHMON process. The name of the PATHCOM object file is $SYSTEM.SYSTEM.PATHCOM.

Start a PATHCOM process by entering the PATHCOM command through the TACL command interpreter and specifying a PATHMON process name.

```
PATHCOM [ / run-option [ , run-option ]... / ]
[ pathmon-name [ ; command [ ; command ]... ] ]


run-option is:


     IN command-file
     OUT list-file
     CPU number
     NAME $process-name
     NOWAIT
     PRI number
```

run-option

is one of the TACL RUN [D] command run options. For details about the RUN [D] run options, see the *TACL Reference Manual*.

IN **command-file**

specifies a file from which commands are read. The command file can contain the Pathway configuration (that is, the PATHCOM commands that establish characteristics for the Pathway environment, the terminals, the TCPs, and the server classes). PATHCOM reads 132-byte records from the specified file until an end-of-file character or an EXIT command is encountered.

If you omit this parameter, input is taken from the current TACL input file; this file is typically the home terminal.

OUT **list-file**

specifies a file to which all PATHCOM command responses are written. The file must be either an EDIT file or an unstructured format file. If responses are written to a key-sequenced, relative, or entry-sequenced file, the Pathway configuration fails and the PATHCOM process eventually abends.

If you specify a disk file name that already exists, the PATHCOM command response is appended to the existing file. If you specify a disk file and the file does not exist, an EDIT file is created.

If you specify a line printer or a process, a page eject is performed when the file is opened. A page eject is not performed before the file is closed.

If you specify a magnetic tape, two consecutive file marks are written just before the file is closed.

If you omit this parameter, output is directed to the current TACL output file; this is typically the home terminal.

CPU **number**

specifies the processor in which PATHCOM runs. Possible values are any processor on your system, from 0 through 15.

If you omit this parameter, the default is the processor used by the TACL command interpreter.

NAME $**process-name**

specifies the name of the PATHCOM process. The name can be a maximum of 6 characters. $

process-name

begins with a dollar sign ($), followed by 1 to 5 alphanumeric characters. The first alphanumeric character must be a letter.

If you intend to use the PATHCOM process name across a network, $

process-name

can be no longer than five characters, that is, a dollar sign ($) followed by no more than four alphanumeric characters.

NOWAIT

specifies that the TACL command interpreter is not to wait for PATHCOM, but is to return immediately with a command input prompt. This option is usually not included when PATHCOM is used interactively, but is useful when PATHCOM is used with a command file.

PRI **number**

is the execution priority of the PATHCOM process.

number

can be a value from 1 through 199. If you omit this parameter, the default is a priority slightly greater than the priority of the PATHMON process.

pathmon-name

specifies the name of the PATHMON process with which PATHCOM is to establish communication. The format is:

[**node.**]$**process-name**

If

node

is omitted, PATHCOM opens the PATHMON process with the same system name as the name of the TACL command interpreter. You can use a system number in place of node. If $

process-name

is omitted, a default name of $PM is used for the first Pathway environment.

command

specifies a PATHCOM command. When commands are present, PATHCOM runs the commands and then terminates. If an IN

command-file

parameter is present, PATHCOM ignores it. Valid commands for this parameter depend on the state of the objects affected by the commands.

Errors

This table lists the most common error that can occur during the processing of the PATHCOM startup command:

This Message…	Is Displayed When…
FILE OPEN ERROR - **pathmon-name** - FILE IN USE (12)	You attempted to start a PATHCOM process but too many processes are running concurrently under the PATHMON process. There is a limit to the number of concurrently running processes (TCPs, external TCPs, ACS subsystem processes, PATHCOMs, and SPI processes) that a PATHMON process can handle. For more information on Pathway configuration limits and defaults, see **Configuration Limits and Defaults** on page 364. Retry the TACL command to start the PATHCOM process; after one of the other processes stops.

PATHCOM Operation Commands

The commands for controlling and operating PATHCOM are:

CMDCWD

CMDVOL

ERRORS

EXIT

FC

HELP

HISTORY

OBEY

OBEYVOL

OPEN

RESET

SHOW

!

NOTE: When using TS/MP 2.6 or later, Hewlett Packard Enterprise recommends that you use PDMCOM instead of PATHCOM, because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

CMDCWD Command

Use the CMDCWD command to set a default OSS directory. When processing ADD or ALTER commands, the PATHMON process uses this default directory to resolve OSS relative file names only if no value is defined for the CWD attribute.

```
CMDCWD oss-pathname
```

oss-pathname

specifies an absolute OSS directory name. For more information on OSS pathnames, see the subsection **OSS Pathnames** on page 137.

Considerations

- When resolving relative pathnames, the PATHMON process first looks for the OSS directory defined for the CWD attribute. If no value is defined, the OSS directory defined by the CMDCWD command is used.
- The CMDCWD and the SET SERVER CWD commands both define a value for the CWD attribute. They do not function the same way, however.

  When resolving relative OSS pathnames, the value defined by the SET SERVER CWD command takes precedence. The RESET SERVER command resets this value to null.

  The value defined by the CMDCWD command is used to resolve relative OSS pathnames only if no value is set using the SET SERVER CWD command. It is a default absolute pathname that you can

define for a server class. The RESET SERVER command has no effect on this value. To reset this value to null, use the RESET CMDCWD command.

- The CMDCWD command operates differently from the CMDVOL command. PATHCOM uses values defined with the CMDVOL command to resolve file names when it processes the SET command. But PATHCOM uses values defined with the CMDCWD command to resolve relative pathnames when it processes ADD or ALTER commands.

  If a default absolute pathname /abc is defined with the CMDCWD command (assuming no value is set for the CWD attribute), all OSS relative path names in ADD or ALTER commands are resolved with the /abc absolute pathname. The resolved path names are sent to the PROCESS_SPAWN_ procedure. However, the default OSS directory name and other relative OSS path names are stored separately in the PATHMON configuration file.

  If you issue the CMDCWD command again to assign the pathname /xyz (and there is still no value defined by the SET SERVER CWD command), all relative path names sent thereafter to PROCESS_SPAWN_ procedure are now resolved with the /xyz pathname.

CMDVOL Command

Use the CMDVOL command to set the default node, volume, or subvolume for the expansion of any Guardian file names associated with objects supported under NonStop TS/MP or Pathway/iTS, except command file names used with the OBEY command. Guardian file names specified in SET commands are expanded with the current value defined by the CMDVOL command.

```
CMDVOL { [ \node. | \*.] $volume.subvolume }
       { [ \node. | \*.] $volume }
       { [ \node. | \*.] subvolume }
       { \node | \* }
```

\node

specifies the node name to be used for file expansion.

*****

specifies that the node where the PATHMON process is running at any given time be used as the node name for file name expansion. * is a generic node name designating PATHMON's current node. This is the default specification when the Pathway NODEINDEPENDENT attribute is set to ON.

$volume

specifies the volume to be used for file expansion.

subvolume

specifies the subvolume to be used for file expansion.

Considerations

- If you omit the CMDVOL command, the default setting in effect is the node, volume, and subvolume of the user who started PATHCOM.
- The first time the CMDVOL command is run, any parameters not specified default to the settings of the user who started PATHCOM. For subsequent CMDVOL commands, parameters not specified default to the value established by the previous CMDVOL command.

  As an example, assume that the first time the CMDVOL command is issued, the volume is not specified. The volume defaults to $CHICAGO, the volume of the user who started PATHCOM. A second CMDVOL command is issued, this time explicitly setting the volume as $PROVO. If a third CMDVOL command is issued without a specified volume, the value for the volume defaults to $PROVO.

- After a CMDVOL command is run, it remains in effect until either another CMDVOL command is run or the PATHCOM session ends. Thus, if the node \MARS is defined with the CMDVOL command, all Guardian file names subsequently defined without specified node name are expanded with the \MARS node name.

  If you issue the CMDVOL command again to assign the node \DALLAS, all subsequent Guardian file names defined without specified node names are expanded, and stored in the PATHMON configuration file, with the \DALLAS node name. Previous file names resolved with the \MARS node are not changed.

- Setting the Pathway NODEINDEPENDENT attribute to ON overrides and disables the node field of the CMDVOL command. If PATHWAY NODEINDEPENDENT is ON and you use CMDVOL to specify a default node other than *, you get error*2075* CMDVOL NODE VALUE CONFLICTS WITH PATHWAY NODEINDEPENDENT ON. Hewlett Packard Enterprise recommends that you must not specify a node field with the CMDVOL command when the NODEINDEPENDENT attribute is ON. For details on the NODEINDEPENDENT attribute of the SET PATHWAY command, see **PATHMON Environment Control Commands** on page 159.
- The CMDVOL command is in effect only for the duration of the current PATHCOM session. Thus, if the NODEINDEPENDENT attribute of the SET PATHWAY command is set to OFF and you use CMDVOL to specify * as the default node for file expansion, the * default is in effect only for the duration of your PATHCOM session. The following table illustrates the effect of different combinations of the CMDVOL command and the NODEINDEPENDENT attribute:

Table 12: Node Value Based on CMDVOL and NODEINDEPENDENT Settings

CMDVOL Node Setting	Node Value When NODEINDEPENDENT Set to ON	Node Value When NODEINDEPENDENT Set to OFF
Node not specified	* (persistent across PATHCOM sessions)	Default of user who started PATHCOM
\node explicitly specified	Error *2075* -- CMDVOL NODE VALUE CONFLICTS WITH PATHWAY NODEINDEPENDENT ON	**\node**
*	* (persistent across PATHCOM sessions)	*(not persistent; must be reset for each PATHCOM session)

Examples

This sequential example show a default set with the CMDVOL command when the Pathway NODEINDEPENDENT attribute is set to OFF. Note that parameters remain in effect until a new CMDVOL command sets a new parameter.

1> CMDVOL $ENGR.SYS1	Expanded filename is \ **current** -PATHCOM- **user- node** .$ENGR.SYS1. **filename**
2> CMDVOL \HQ. $MKT.SALES	Expanded filename is \HQ.$MKT.SALES. **filename**
3> CMDVOL $ADMIN	Expanded filename is \HQ.$ADMIN.SALES. **filename**
4> CMDVOL SYS2	Expanded filename is \HQ.$ADMIN..SYS2. **filename**
5> CMDVOL \FIELD.$SUPP	Expanded filename is \FIELD.$SUPP..SYS2. **filename**
6> CMDVOL *. $ADMIN.VAC	Expanded filename is \ **filename** -PATHMONnode.$ ADMIN.VAC. **filename**

where

current-PATHCOM-user-node

is the node of the user who started PATHCOM.

This example shows a default set when the Pathway NODEINDEPENDENT attribute is set to ON:

CMDVOL *	Expanded filename is \ **current** -PATHMON- **node. current-PATHCOM- user-volume.current- PATHCOM- user-subvolume.filename**

where

current-PATHCOM-user-node

is the node of the user who started PATHCOM.

ERRORS Command

Use the ERRORS command to set the number of errors and warnings that PATHCOM ignores when executing an IN or OBEY command file.

```
ERRORS [ number ]
```

number

specifies the number of errors and warnings that PATHCOM ignores. You can specify a number from -1 to 32767 inclusive where -1 causes PATHCOM to allow an infinite number of errors, 0 or no number causes PATHCOM to stop after encountering the first error, and a positive integer causes PATHCOM to stop after encountering the number of errors specified. If no value is specified, the default value is used; the default value is 0 (zero).

Considerations

- The ERRORS command must appear in the command-line before any OBEY command; otherwise, PATHCOM ignores the ERRORS command and stops after the first error or warning.
- You can enter the ERRORS command in the command file that you use for the IN file option of the PATHCOM startup command; however, if you enter ERRORS on the same command-line with the IN file option, PATHCOM ignores the IN command file to run the ERRORS command.

Examples

This command specifies that PATHCOM read the command file PCCMD until it encounters a total of more than 5 errors and warnings or the end of the file:

```
=ERRORS 5;OBEY PCCMD
```

This command specifies that PATHCOM read the command file PWCMD until it encounters a total of more than 10 errors and warnings or the end of the file:

```
PATHCOM $PM1;ERRORS 10;OBEY PWCMD
```

EXIT Command

Use the `EXIT` command to terminate communication with PATHCOM. If PATHCOM is using a command file, this command terminates the command file.

```
EXIT
```

PATHCOM stops when it encounters the `EXIT` command in a command file (the IN file).

FC Command

Use the `FC` command to edit or repeat a command line.

```
FC [ [-] number | string ]
```

number is the command number of the command to be retrieved. If number is passed as -7, the 7th previous command is retrieved.
string is the first character of a previous command.

This command displays the previous command line and prompts to edit input with a period (.). FC accepts three subcommands:

R(**replacement-string**)	Replaces one or more characters.
I (**insertion-string**)	Inserts one or more characters.
D (**deletion-character**)	Deletes one character.

Considerations

- You enter subcommands and their associated strings beneath the display command-line and terminate these commands with a carriage return. Begin replacement, insertion, and deletion commands with the character positioned directly above the subcommand (R,I,D).

Subcommand R replaces characters in the command-line with **replacement-string** on a one-for-one basis. Subcommand I inserts **insertion-string** in the command-line before the character that is above **insertion-string**. Subcommand D deletes the character that is above it in the command-line; you can repeat the subcommand for each character to be deleted. If you enter a string of characters but no command, replacement R occurs.

- After you edit the command-line, entering FC displays the command-line and prompts for another subcommand. FC terminates when it receives only a carriage return; the corrected command-line is then run. You can terminate the FC command without execution by pressing BREAK, pressing CTRL/Y, or entering a double slash (//) in columns 1 and 2, immediately followed by a carriage return.
- You can separate FC subcommands by entering the double slash. For example:

```
SET PATHWAY MAXXTERMS 45
d// r50
```

The corrected command is:

```
SET PATHWAY MAXTERMS 50
```

- You can specify a string after the FC command. If the string matches the starting letters of any of the prior ten commands, the corresponding command appears. It also allows you to edit and execute the command.
- The FC command is displayed only if the command length is greater than 2.

Example

- In this example, entering each subcommand followed by a carriage return causes the indicated changes to the command-line:

```
SET MAXXTERMS 45
      d           <---- deletes the extra letter X
   iPATHWAY       <---- inserts the word PATHWAY
              r50 <---- replaces the number 45 with 50
```

and displays the corrected command:

```
SET PATHWAY MAXTERMS 50
```

- If the following commands are entered at the PATHCOM prompt:

```
\NSSYS.$VOL.SUBVOL 11> PATHCOM
$X30Q: PATHCOM - T0845H02 - (30APR09)
(C)2008-2009 Hewlett Packard Development Company, L.P.
OPEN $PM
^
$X30Q: ERROR - FILE OPEN ERROR - $PM - nonexistent device
(14)
= [ SERVER SRV1 is configured in $PM1
= OPEN $PM1
= ALTER SRV1, NUMSTATIC 1
= ALTER SRV1, DEBUG ON
```

To edit and execute the fourth command using a number, enter:

```
= FC 4
```

The fourth command appears as:

```
ALTER SRV1, NUMSTATIC 1
. <enter>
=
```

To edit and execute a command using **string**, enter:

```
= FC AL
```

The command starting with “AL” that is executed last, is displayed:

```
= ALTER SRV1, NUMSTATIC 1
```

HELP Command

Use the HELP command to display or list to a file the command syntax for all PATHCOM commands.

```
<informaltable frame="all">
<tgroup char="" charoff="50" cols="1">
<?PubTbl tgroup rth="1.00pt"?>
<colspec colname="col1" colwidth="1.00*"/>
<tbody>
<row valign="top"><entry></entry></row>
</tbody>
</tgroup>
</informaltable>
```

OUT **list-file**

specifies the name of a file to which PATHCOM directs the HELP output. If you omit this option, the output goes to the PATHCOM list file; this is typically the home terminal.

ALL

lists the syntax for all PATHCOM commands.

COMMANDS

lists the names of all PATHCOM commands.

<command-name>

specifies a particular PATHCOM command syntax to appear. When entering the name of the command, do not include the angle brackets.

<“**detail-name**”>

specifies a particular parameter syntax to appear. When entering the name of the parameter, include the angle brackets; do not include the quotation marks.

<**error-num**>

specifies a Pathway environment error number from 1000 through 3999 for which an explanation is displayed. When entering the number of the error, do not include the angle brackets.

Display Format

The HELP command initially displays this syntax for the HELP command and lists the command names for which help is available:

```
HELP [ / OUT list-file / ] [ ALL ]
[ COMMANDS ]
[ <command-name> ]
[<"detail-name"> ]
[ <error-num> ]
```

command-name ::=

ABORT	ADD L	ALTER	CMDCWD	CMDVO
CONTROL	DELETE	ERRORS	EXIT	FC
FREEZE	HELP	INFO	INSPECT	LOG1
LOG2	O	OBEY	OBEYVOL	OPEN
PRIMARY	REFRESH-CODE	RESET	RESUME	RUN
SET	SHOW	SHUTDOWN	SHUTDOWN2	START
STATS	STATUS	STOP	SUSPEND	SWITCH
TELL	THAW			

Examples

The command HELP SUSPEND, PATHCOM displays the SUSPEND command syntax as:

```
SUSPEND <termname-list> [ ! ]
```

To next obtain syntax for the <termname-list> parameter of the SUSPEND command, enter:

```
HELP <termname-list>
```

In response, PATHCOM displays the syntax for a *TERM* parameter list as:

```
<termname-list> ::=
TERM * [ [ , <termqualifier> ] ... ]
[ TERM ] <name-list>
```

To obtain syntax for the <termqualifier> parameter of the SUSPEND command, enter:

```
HELP <termqualifier>
```

In response, PATHCOM displays the syntax for a TERM qualifier as follows:

```
<termqualifier> ::=
{ SEL } [ NOT ] { RUNNING }
{ STATE } { STOPPED }
{ SUSPENDED }
TCP <name>
```

Entering an error number with the HELP command as follows:

```
HELP 3019
```

causes PATHCOM to display the corresponding message text:

```
PATHTCP ERROR - *3019* WRONG TRANSFER COUNT IN TERMINAL IO
```

HISTORY Command

Use the HISTORY command to display previous PATHCOM commands.

```
HISTORY [ number ]
```

number

is the number of earlier commands you want to retrieve. The default value is 10.

Consideration

- If you omit **number**, the HISTORY command displays the last ten commands.
- If **number** is greater than the number of commands issued previously, the HISTORY command displays all the commands.
- The HISTORY command displays the command numbers for each command. You can use command numbers in the FC and ! commands.
- The HISTORY command can display a maximum of ten commands.

Examples

- When five commands are entered at the PATHCOM prompt:

```
\NSSYS.$VOL.SUBVOL 11> PATHCOM
$X30Q: PATHCOM - T0845H02 - (30APR09)
(C)2008-2009 Hewlett Packard Development Company, L.P.
= [ SERVER SRV1 is configured in $PM1
= OPEN $PM1
= ALTER SRV1, NUMSTATIC 1
= ALTER SRV1, DEBUG ON
= ALTER SRV1, HOMETERM $ZTNT.#PTA2AK3
= ALTER SRV1, PROCESS $PROC
```

- To display the last three commands, enter:

```
= HISTORY 3
```

- In response, PATHCOM displays the following output:

```
3>ALTER SRV1, DEBUG ON
4>ALTER SRV1, HOMETERM $ZTNT.#PTA2AK3
5>ALTER SRV1, PROCESS $PROC
```

OBEY Command

Use the OBEY command to cause commands to be read from a specified command file.

```
{ OBEY } file-name
{ O    }
```

file-name

specifies a Hewlett Packard Enterprise file name. The file can be a disk file, terminal, or process from which PATHCOM reads commands. **file-name** can also be the name of a DEFINE.

Considerations

- PATHCOM reads and runs commands from the named file until it encounters an EXIT command or the end of the file. After the end of the file, PATHCOM closes the command file and command input reverts to the file from which the OBEY command was read. Multiple OBEY commands can appear within a command file; you can nest command files to a depth of four.
- If PATHCOM detects an error while processing the commands from a command file, it closes that file and any other command files currently open (in the case of nested command files). If the original input file for PATHCOM is a terminal, PATHCOM displays a prompt on the terminal. If the input file is not a terminal, PATHCOM terminates.

Examples

This command reads and runs the commands contained in the file MFG.CMDSFILE. In this example, both the subvolume and the file identifier are specified:

```
OBEY MFG.CMDSFILE
```

This command also reads and runs the commands contained in the file CMDSFILE. However, because no subvolume was specified, PATHCOM looks for the file in the subvolume from which you issued the command:

```
OBEY CMDSFILE
```

In this example, the file name is a DEFINE that you specify before starting PATHCOM:

```
12> ADD DEFINE =CMD, CLASS MAP, FILE $DATA.PW.CONFIG2
13> PATHCOM $PM1
= OBEY =CMD
```

OBEYVOL Command

Use the OBEYVOL command to set the default node, volume, and subvolume for expansion of OBEY command file names.

```
OBEYVOL [ [ \node.] $volume.subvolume ]
        [ [ \node.] $volume ]
        [ [ \node.] subvolume ]
        [ \node ]
```

\node

specifies the node name to be used.

$volume

specifies the volume to be used.

subvolume

specifies the subvolume to be used.

Considerations

- If the node, volume, and subvolume are omitted and the current command file is a disk file, this command sets the default settings to that of the current command file.
- If the OBEYVOL command is not issued, the default setting is the node, volume, and subvolume in which PATHCOM is running.

Examples

This command sets the default node, volume, and subvolume:

```
OBEYVOL \NY.$MKT.ABC
```

This command sets only a default volume and subvolume; the node automatically defaults to the one in which PATHCOM is running:

```
OBEYVOL $ENGR.DEF
```

OPEN Command

Use the OPEN command to specify the name of the PATHMON process—and therefore the Pathway environment—to which PATHCOM directs subsequent commands. This command is useful when more than one Pathway environment is running on the same NonStop system. You can direct commands to multiple PATHMON processes during a single PATHCOM session.

```
OPEN [ \node.]$pm-process
```

\node

specifies the Hewlett Packard Enterprise node name or node number on which a PATHMON process can be opened. If this parameter is omitted, PATHCOM opens the PATHMON process with the same node name as itself.

$**pm-process**

specifies the name of the PATHMON process to be opened.

Errors

This table lists the most common error that can occur during the processing of the OPEN command:

This Message…	Is Displayed When…
FILE OPEN ERROR - **pathmon-name** - FILE IN USE (12)	You attempted to start a PATHCOM process but too many processes are running concurrently under the PATHMON process. The number of concurrently running processes (TCPs, external TCPs, ACS subsystem processes, PATHCOMs, and SPI processes) that a PATHMON process can handle is limited. For more information on Pathway configuration limits and defaults, see **Configuration Limits and Defaults** on page 364. Retry the command to start the PATHCOM process; after one of the other processes stops, the PATHMON process can start the PATHCOM process.

Examples

This command opens the PATHMON process named $PATH:

```
OPEN $PATH
```

This command opens the PATHMON process named $PATH3 running on node \SYS1:

```
OPEN \SYS1.$PATH3
```

RESET CMDCWD Command

Use the `RESET CMDCWD` command to change to blanks the default OSS directory name you defined with the `CMDCWD` command.

```
RESET CMDCWD
```

Consideration

This command does not change the OSS directory name for SERVER objects already added to the PATHMON configuration file. It only resets the default OSS directory name used to resolve relative OSS pathnames when a server class is added to the PATHMON configuration file and no value for the CWD attribute is set using the `SET SERVER CWD` command.

SHOW Command

Use the `SHOW` command to display parameter settings that describe the PATHCOM environment. These settings are specific to the PATHCOM session.

You can also use the `SHOW` command to display the current values for attributes of SERVER objects, (and of Pathway/iTS TCP, TERM, and PROGRAM objects as well). For details about the syntax and usage of the `SHOW` command for each of the object types, see the sections on commands for these objects.

```
SHOW [ / OUT list-file / ] { CMDCWD }
                           { CMDVOL }
                           { ERRORS }
                           { OBEYVOL }
```

OUT **list-file**

specifies the name of the list file to which PATHCOM directs output; this can be a DEFINE name. If you omit this option, the output goes to the PATHCOM list file; this file is typically the home terminal.

CMDCWD

displays the default OSS directory name defined with the CMDCWD command. This value is used to resolve relative pathnames specified for OSS server processes if no value is defined using the SET SERVER CWD command.

CMDVOL

displays the default node, volume, and subvolume for expansion of any file names except OBEY command file names. You define this value using the CMDVOL command.

ERRORS

displays the number of errors PATHCOM ignores when reading a command file. This number is set by the ERRORS command, described previously in this section.

If PATHCOM has not yet read the OBEY command file, this option displays the number of errors to be ignored. If PATHCOM has read the file, this option displays the number of errors remaining of the number you specified for PATHCOM to ignore.

OBEYVOL

displays the default node, volume, and subvolume for expansion of command file names.

Examples

The command SHOW CMDVOL displays:

```
CMDVOL \TNS.$DICS.DATA
```

If you used the CMDVOL command to direct that unspecified node names resolve to the node where the PATHMON process is running, or if the Pathway NODEINDEPENDENT attribute is set to ON, the command SHOW CMDVOL displays:

```
CMDVOL \*.$DICS.DATA
```

The command SHOW OBEYVOL displays:

```
OBEYVOL \TNS.$DATA.POOL
```

This command:

```
ERRORS 5
```

specifies that PATHCOM is to ignore up to 5 errors in an OBEY command file.

These commands cause PATHCOM to display the number of errors that remain of the 5 you specified with the ERRORS command:

```
OBEY PCMDS
SHOW ERRORS
```

If PCMDS contained one error, PATHCOM then displays:

```
ERRORS 4
```

! Command

Use the ! command to execute an existing command.

```
! [ [-]number | string ]
```

number

is the number of a command line. If **number** is passed as -7, the 7th previous command is executed.

string

is the first character or characters of a previous command.

Considerations

- The ! command executes the command indicated by the `number`. By default, the last command is processed.
- You can specify a string after the ! command. If the string matches the starting letters of any of the prior ten commands, the corresponding command is executed.
- If you use the ! command without a number or text string, PATHCOM executes the last command you entered.

Example

- If the following commands are entered at the PATHCOM prompt:

```
\NSSYS.$VOL.SUBVOL 11> PATHCOM
$X30Q: PATHCOM - T0845H02 - (30APR09)
(C)2008-2009 Hewlett Packard Development Company, L.P.
= [ SERVER SRV1 is configured in $PM1
= OPEN $PM1
= ALTER SRV1, NUMSTATIC 1
= ALTER SRV1, DEBUG ON
```

- To execute the second command, enter:

```
= !2
```

- The second command is executed as:

```
= ALTER SRV1, NUMSTATIC 1
```

- To execute a command starting with “op”, enter:

```
= !op
```

  The following command is executed:

```
= OPEN $PM1
```

PATHMON Environment Control Commands

This section describes the PATHCOM commands that control the PATHMON process and the PATHMON environment as a whole; the commands are listed in alphabetic order. The PATHMON environment control commands are as follows:

To Perform These Tasks…	Use These PATHCOM Commands…
Define attribute values	SET PATHWAY SET PATHMON
Change attribute values	CONTROL PATHMON
Change process states	START PATHWAY STOP PATHMON SHUTDOWN SHUTDOWN2
Obtain information about	INFO PATHWAY INFO PATHMON STATUS PATHWAY STATUS PATHMON STATUS LINKMON
Perform other management tasks	LOG1 LOG2 PRIMARY PATHMON SWITCH PATHMON

These commands define the PATHMON configuration and control basic PATHMON environment operations. With these commands you can:

- Start and stop PATHMON environment operations
- Designate and change the processor where the PATHMON backup process runs
- Set the limits for the number of PATHMON-controlled objects
- Specify the PATHMON process logging requirements
- Display PATHMON object attributes and status information

The only PATHCOM command for the ACS subsystem is STATUS LINKMON. For more information about ACS subsystem processes, see the **Introduction to NonStop Transaction Processing**. For a list of PATHCOM commands that you to perform functions related to ACS subsystem processes, see **Overview of PATHCOM** on page 130. For detailed information about ACS subsystem processes and their use with Pathsend requestors, see the *TS/MP 2.5 Pathsend and Server Programming Manual*.

NOTE: When using TS/MP 2.6 or later, Hewlett Packard Enterprise recommends that you must use PDMCOM instead of PATHCOM, because it can communicate with multiple PATHMONs simultaneously. For more information on PDMCOM, see the *TS/MP 2.7 ACS Reference Manual*.

CONTROL PATHMON Command

Use the `CONTROL PATHMON` command to change specific attributes of the PATHMON process while it is running and to record the changes in the PATHMON configuration file.

```
CONTROL PATHMON , pathmon-attribute [ , pathmon-attribute
]...

pathmon-attribute is:
    BACKUPCPU number
    DUMP { ON [ ( FILE file-name ) ] | OFF }
    DUMPMEMORY (FILE file-name )
    SPREBALANCECPU number
```

BACKUPCPU **number**

specifies the processor where the PATHMON backup process runs. The PATHMON process stops the existing backup process and creates a new backup process in the processor that you specify.

DUMP {ON | OFF}

specifies whether the PATHMON process writes the data stack information to a file if it encounters an internal error. The default value is OFF.

ON

directs PATHMON to write the information in the data stack to a file and then to set this option to OFF. ON is the Hewlett Packard Enterprise recommended setting. (The only reason not to set this attribute to ON is because of the disk space required if large number of dumps occur.)

FILE file-name

specifies the name of the file that the PATHMON process creates for its dump operation.

If the NODEINDEPENDENT attribute of the SET PATHWAY command is set to ON and you do not specify the node portion of **file-name** , the node name defaults to a generic node name, *, that designates whatever node the PATHMON process is running on after cool start.

If the NODEINDEPENDENT attribute of the SET PATHWAY command is set to OFF and you do not specify the node portion of **file-name** , the node name defaults to the node the PATHMON process is running on when you issue the CONTROL command

If you do not specify the volume and subvolume portions of **file-name** , the PATHMON process creates a dump file on the same subvolume as the PATHMON configuration file. The format of the file name is ZZPM**nnn**, where **nnn** is a 3-digit number. The file code is 130.

NOTE: file-name specifies a file that already exists and the PATHMON process attempts to dump to this file, a dump is not performed and error *1123* PATHMON, MEMORY DUMP NOT TAKEN is generated.

OFF

directs the PATHMON process not to write data stack to a file when it encounters an internal error.

DUMPMEMORY (FILE **file-name**)

Forces the PATHMON process to write the information in its data stack to a file. Note that 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 internal state of the PATHMON process at a particular time—before it encounters a fatal error. You must force a dump only if your representative specifically requests it.

FILE **file-name**

specifies the name of the file that the PATHMON process creates for its dump operation.

If you do not specify **file-name** , the PATHMON process creates a dump file on the same subvolume as the PATHMON configuration file. The format of the file name is ZZPM**nnn** , where **nnn** is a 3-digit number. The file code is 130.

SPREBALANCECPU number

instructs PATHMON to start rebalancing server processes that were originally in the specified CPU number. PATHMON rejects this command if it is not configured with SPREBALANCEMODE MANUAL.

The number indicates the CPU number on the local system in which the PATHMON process is running, and into which the server processes are to be rebalanced.

The command accepts processor number as a parameter. The server processes which used to run on the mentioned processor before processor failure, will be shifted back to the original processor. To validate this command the CPU number must be between 0 -15 and PATHMON must be configured to run in MANUAL mode (SPREBALANCEMODE).

Considerations

- If the primary PATHMON process is running in the processor defined for the backup PATHMON process because of a `SWITCH` command or some other processor change, the `BACKUPCPU` option causes the PATHMON process to change the PATHMON configuration file so that it contains the current processor numbers for both the primary and backup PATHMON processes.
- When you cool start a Pathway environment (that is, when you use a pre-existing configuration file to start a Pathway environment) after using the `CONTROL PATHMON` command, the PATHMON process uses the values you changed with the `CONTROL PATHMON` command.
- When the creation of a backup process fails more than ten times, the PATHMON process delays trying to create a backup process. The delay is 1 minute if stopped 11 times, 2 minutes if stopped 12 times, and so forth. However, the PATHMON process does not delay creating a backup process if any of these occur:

  - The processor in which the PATHMON primary or backup process is executing fails.
  - A `PATHCOM SWITCH` command is run.
  - A `CONTROL PATHMON BACKUPCPU` command is run.

  In addition to not delaying the creation of a backup process if the preceding conditions occur, the PATHMON process resets its internal counter. Consequently, creation of a backup process has to fail again 10 more times before the PATHMON process delays creating the backup process.

- Hewlett Packard Enterprise recommends that DUMP is set to ON for production systems. Having a DUMP file available greatly enhances your ability to solve system problems, and is a likely requirement when requesting help from your Hewlett Packard Enterprise representative.
- If you encounter system problems and the PATHMON process has not created a dump (because DUMP was not set to ON), use the `DUMPMEMORY` option to force a dump.

Errors

This table lists the most common errors that can occur during the processing of the `CONTROL PATHMON` command:

This Message…	Is Displayed When…
1168 REBALANCE PROCESSOR DOWN	The processor number mentioned in the CONTROL PATHMON SPREBALANCECPU command is down, the rebalancing of the server processes cannot be started and the PATHMON process returns this error.
1093 BACKUP PROCESSOR DOWN	For a multiprocessor system, you specified a backup processor 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 processor in which the PATHMON primary process is running. The location of the backup process is not changed in this case. OR You specified for the SPREBALANCECPU attribute, the number that has the first character as digit and remaining are alphanumeric or special characters.

Example 1

This command creates a backup PATHMON process in processor 5:

```
CONTROL PATHMON, BACKUPCPU 5
```

This command changes the `PATHMON DUMP` option to OFF:

```
CONTROL PATHMON, DUMP OFF
```

This command creates a backup PATHMON process in processor 4, and creates a dump file named PMDUMP if an internal error occurs:

```
CONTROL PATHMON, BACKUPCPU 4, DUMP ON (FILE PMDUMP)
```

Example 2

```
CONTROL PATHMON, SPREBALANCECPU <CPUNO>
```

NOTE: `<CPUNO>` is the processor number of the system where PATHMON process is running.

INFO PATHMON Command

Use the `INFO PATHMON` command to display the attribute values currently defined for the PATHMON process.

```
INFO [ / OUT list-file / ] PATHMON [ , OBEYFORM ]
```

OUT **list-file**

directs output to the named list file; this can 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.

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.

Consideration

OBEYFORM is an INFO command option for the PATHMON process, PATHWAY, SERVER objects, as well as for the Pathway/iTS TCP, TERM, 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 Pathway environment configuration. You must add the START PATHWAY command at the end of the OUT file before using the file to start a Pathway environment. See the examples in this subsection.

Examples

The INFO PATHMON command causes the PATHMON process to display:

```
=INFO PATHMON
PATHMON
BACKUPCPU 4
DUMP ON (FILE \SY.$VOL1.TESTING.MONDUMP)
```

This command:

```
INFO PATHMON, OBEYFORM
```

causes PATHMON to display the information in command form:

```
SET PATHMON BACKUPCPU 4
SET PATHMON DUMP ON (FILE \SY.$VOL1.TESTING.MONDUMP)
```

This command directs the information to a file:

```
INFO/OUT PMINFO/PATHMON, OBEYFORM
```

These commands create a DEFINE at PATHCOM startup and direct the information to the file specified in the DEFINE:

```
12> ADD DEFINE =OUT-LOG, CLASS SPOOL, LOC $S, REPORT "INFO"
12> PATHCOM $PM1

=INFO/ OUT =OUT-LOG/SERVER *
```

This sequence of commands captures a configuration for a Pathway environment (minus the START PATHWAY command) in the command file called NEWCONFG:

```
=INFO/OUT NEWCONFG/PATHMON, OBEYFORM
=INFO/OUT NEWCONFG/PATHWAY, OBEYFORM
=INFO/OUT NEWCONFG/SERVER *, OBEYFORM
```

NOTE:

The configuration in this example does not include requestor objects that run under Pathway/iTS, although if configured, INFO PATHMON command displays these objects.

INFO PATHWAY Command

Use the INFO PATHWAY command to display both the PATHMON-controlled object attribute values you defined using the SET command, and the values currently in effect. For object attributes not yet defined, this command displays the default value or a question mark character. This command displays Pathway attributes SPREBALANCEMODE and DUMASK. These attributes are not mandatory attributes. However, the INFO PATHWAY command displays these attributes only if they are explicitly configured.

```
INFO [ / OUT list-file / ] PATHWAY [ , OBEYFORM ]
```

The INFO PATHWAY command lists the default Pathway parameters as follows:

```
Pathway
  MAXASSIGNS N                    [CURRENTLY M]
  MAXDEFINES N                    [CURRENTLY M]
  MAXEXTERNALTCPS N               [CURRENTLY M]
  MAXLINKMONS N                   [CURRENTLY M]
  MAXPARAMS N                     [CURRENTLY M]
  MAXPATHCOMS N                   [CURRENTLY M]
  MAXPROGRAMS N                   [CURRENTLY M]
  MAXSERVERCLASSES N              [CURRENTLY M]
  MAXSERVERPROCESSES N            [CURRENTLY M]
  MAXSPI N                        [CURRENTLY M]
  MAXSTARTUPS N                   [CURRENTLY M]
  MAXTCPS N                       [CURRENTLY M]
  MAXTELLQUEUE N
  MAXTELLS N                      [CURRENTLY M]
  MAXTERMS N                      [CURRENTLY M]
  MAXTMFRESTARTS N
  OWNER \OCOOL.G,U
  SCLONGPROCESSNAME ON
  SECURITY "N"
```

OUT **list-file**

directs output to the named list file; this file can 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.

OBEYFORM

causes the information to appear in the format you would use to set up a PATHMON configuration file. Each attribute is displayed as a syntactically correct PATHCOM command with the proper SET prefix.

Consideration

OBEYFORM is an INFO command option for the PATHMON process, PATHWAY, and SERVER objects, as well as for the Pathway/iTS TCP, TERM, 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 Pathway configuration. You must add the START PATHWAY command at the end of the OUT file before using the file to start a Pathway environment.

Examples

After you define the attribute values with the SET PATHWAY command, you can use the INFO PATHWAY command to display those values. This example shows the output from an INFO PATHWAY command after the attribute values are defined:

```
PATHWAY
   DUMASK 007
   GLINKALGORITHM "REQUESTERCPU"
   MAXASSIGNS 1200                  [CURRENTLY 759]
   MAXDEFINES 50                    [CURRENTLY 5]
   MAXEXTERNALTCPS 0                [CURRENTLY 0]
   MAXLINKMONS 16                   [CURRENTLY 0]
   MAXPARAMS 30                     [CURRENTLY 2]
   MAXPATHCOMS 10                   [CURRENTLY 1]
   MAXPROGRAMS 5                    [CURRENTLY 0]
   MAXSERVERCLASSES 350             [CURRENTLY 320]
   MAXSERVERPROCESSES 1750          [CURRENTLY 1544]
   MAXSPI 5                         [CURRENTLY 0]
   MAXSTARTUPS 400                  [CURRENTLY 0]
   MAXTCPS 5                        [CURRENTLY 0]
   MAXTELLQUEUE 4
   MAXTELLS 32                      [CURRENTLY 0]
   MAXTERMS 20                      [CURRENTLY 0]
   MAXTMFRESTARTS 5
   NODEINDEPENDENT ON
   OWNER \*.255,63
   SCLONGPROCESSNAME ON
   SECURITY "U"
   SPREBALANCEMODE “DISABLE”
```

NOTE:

- The configuration in this example does not include requestor objects that run under Pathway/iTS, if configured, the INFO PATHWAY command these objects.
- GLINKALGORITHM is made available for all L-series RVUs

This example shows the output from an INFO PATHWAY, OBEYFORM command before the attribute values are defined:

```
SET PATHWAY DUMASK 007
SET PATHWAY GLINKALGORITHM "REQUESTERCPU"
SET PATHWAY MAXASSIGNS ?
SET PATHWAY MAXDEFINES 0
SET PATHWAY MAXLINKMONS 0
SET PATHWAY MAXPARAMS ?
SET PATHWAY MAXPATHCOMS 5
SET PATHWAY MAXSERVERCLASSES ?
SET PATHWAY MAXSERVERPROCESSES ?
SET PATHWAY MAXSPI 1
SET PATHWAY MAXSTARTUPS ?
SET PATHWAY NODEINDEPENDENT OFF
```

```
SET PATHWAY OWNER \TS.8,8
SET PATHWAY SCLONGPROCESSNAME ON
SET PATHWAY SECURITY "U"
SET PATHWAY SPREBALANCEMODE “DISABLE”
```

These commands direct the Pathway environment information to a file:

```
INFO/OUT PWINFO/PATHWAY
INFO/OUT PWINFO/PATHWAY, OBEYFORM
```

Note that you must add the `START PATHWAY` command to the OUT file before using the file to start a Pathway environment.

LOG1 and LOG2 Commands

Use the `LOG1` and `LOG2` commands to name the files that the PATHMON process and the Pathway/iTS TCPs use for reporting errors and changes in object status. The PATHMON process generates either text or event messages for each log file; you specify which type of message the PATHMON process generates. Event messages are used for reporting to an operator or application program through the Event Management Service (EMS), which is a part of Distributed Systems Management (DSM). For more information about Pathway environment event messages, see the *TS/MP Management Programming Manual* and the *Pathway /TS Management Programming Manual*.

The PATHMON process records whether logging is d or disabled in the PATHMON configuration file. After a cool start (that is, a start which uses an already existing PATHMON configuration file), the log files are set to the state indicated in the PATHMON configuration file.

Two logging commands are provided (that is, LOG1 and LOG2) to two copies of the information.

```
LOG1 [ file-name [ , logparam [ , logparam ] ] ]
LOG2 [ file-name [ , logparam [ , logparam ] ] ]
logparam is:
    STATUS
    EVENTFORMAT
```

file-name

specifies the name of a file to receive error reports and changes in status. If you specify a disk file, the file must exist. You must create the file, by using the command interpreter or the FUP CREATE command, before issuing a LOG1 or LOG2 command.

You need not specify a fully-qualified file name for a log file. If you leave the node name unspecified, LOG1 and LOG2 will always run on the node where the PATHMON process is running. If you explicitly specify the node name as * (a generic name that indicates the node where the PATHMON process is running at any given time), PATHCOM will accept the * specification. However, when you use the STATUS or INFO commands to display information about your environment, the * is not displayed as part of the log file name. If you omit this attribute, any current log file is closed.

CAUTION: If you turn off logging by closing a current log file, all internally queued messages are lost.

STATUS

sends status change messages and error messages to the file you specify with **file-name** . If the STATUS option is not specified, only error messages are written to the log file.

EVENTFORMAT

specifies that messages must be formatted as event messages. If you omit EVENTFORMAT, text messages are generated.

Considerations

- The OUT file initially specified with the PATHMON RUN command becomes the initial LOG1 file until you specify otherwise by issuing a logging command. The file is opened when you specify the STATUS attribute.
- As a best practice, specify a disk file for logging than a terminal because log messages are lost after they scroll off the terminal screen. Also, performance is generally better to a disk file than to a terminal.

  If the logging file is a terminal, as a best practice, use that terminal only for logging. If that is not possible, restrict the activity on that terminal to avoid interfering with messages from the PATHMON process. In either case, type “pause” at the TACL prompt to suspend the TACL process so that logging can occur.

  NOTE: If a logging terminal is unavailable, the PATHMON environment might slow down and be unable to support transaction processing.

- EDIT files are not acceptable log files to the PATHMON process. If you attempt to use an EDIT file for a log, the PATHMON process returns an error and fails to open the log.
- If a logging command is used with no attributes, the corresponding log file is closed.
- Logging of status messages is automatically disabled during execution of the SHUTDOWN2 command.

Examples

This command sends error information to a terminal:

```
LOG1 $TERM1
```

This command logs errors and status change messages in a file named COMPLOG:

```
LOG2 COMPLOG,STATUS
```

This command logs errors and status change messages to $0, and formats the messages as event messages:

```
LOG1 $0, EVENTFORMAT, STATUS
```

This command closes the LOG2 file:

```
LOG2
```

PRIMARY PATHMON Command

Use the PRIMARY PATHMON command to resume operation of the PATHMON primary process in the processor defined for this process in the PATHMON configuration file. Use this command after an individual process or processor failure, or after a SWITCH command has moved the PATHMON primary process from one processor to another.

```
PRIMARY PATHMON [ , IFPRICPU number ]
```

IFPRICPU **number**

specifies the processor in which the PATHMON primary process is to run. If this processor is defined for the PATHMON process in the PATHMON configuration file at the time that you enter this command, the PATHMON process resumes the operation of its primary process in this processor.

If the PATHMON primary process is already running in this processor, or if this is not the processor configured for the PATHMON primary process, the PATHMON process ignores the command.

If you omit this option, the PATHMON process resumes its primary process in the processor recorded in the PATHMON configuration file. If this processor is down, no error occurs.

Considerations

- After a series of SWITCH and CONTROL commands, the PATHMON primary process might be running in a processor other than the one you initially defined. Use the STATUS PATHMON command to display the CPUs where the primary and backup PATHMON processes are running.

  If the PATHMON primary process is not running in its configured processor, this command causes the primary process to run in that processor.

- After successfully executing a PRIMARY command, the PATHMON process logs this status message:

  ```
  STATUS - *1077* TAKEOVER BY BACKUP
  ```

Examples

This command resumes the operation of the PATHMON primary process in the processor defined in the PATHMON configuration file:

```
PRIMARY PATHMON
```

This command resumes the operation of the PATHMON primary process in processor 2 only if processor 2 is the processor recorded in the PATHMON configuration file for the PATHMON process.

```
PRIMARY PATHMON, IFPRICPU 2
```

SET PATHMON Command

Use the SET PATHMON command to establish the values for the PATHMON attributes. Use this command before cold starting a Pathway environment.

```
SET PATHMON pathmon-attribute [ , pathmon-attribute ]...

pathmon-attribute is:

    BACKUPCPU number
    DUMP { ON [ ( FILE file-name ) ] | OFF }
```

BACKUPCPU **number**

specifies the processor in which the PATHMON backup process runs.

DUMP {ON | OFF}

specifies whether the PATHMON process writes its data stack information and the contents of the PATHMON configuration file to a file if the PATHMON process encounters an internal error. The default is OFF.

ON

directs the PATHMON process to write the information in its data stack and the contents of the PATHMON configuration file to a file and then to set this option to OFF.

FILE file-name

specifies the name of the file that the PATHMON process creates for its dump operation.

If you do not specify **file-name** , the PATHMON process creates a dump file on the same subvolume as the PATHMON configuration file. The format of the file name is ZZPM**nnn** , where **nnn** is a 3-digit number. The file code is 130.

NOTE: **file-name** specifies a file that already exists and the PATHMON process attempts to dump to this file, a dump is not performed and error *1123* PATHMON, MEMORY DUMP NOT TAKEN is generated.

OFF

directs the PATHMON process not to write data stack and PATHMON configuration file information to a file when it encounters an internal error.

Errors

This table lists the most common errors that can occur during the processing of the SET PATHMON command:

This Message…	Is Displayed When…
1093 BACKUP PROCESSOR DOWN	For a multiprocessor system, you specified a backup processor but only one processor is operational. For a single processor system, you included the BACKUPCPU attribute.
1095 ILLEGAL CPU NUMBER	You specified for the BACKUPCPU attribute the same processor in which the PATHMON primary process is running. The location of the backup process is not changed in this case.

Examples

This command defines processor 3 as the processor for the PATHMON backup process:

```
SET PATHMON BACKUPCPU 3
```

This command defines processor 4 as the backup processor and specifies a dump file named PMDUMP:

```
SET PATHMON BACKUPCPU 4, DUMP ON (FILE PMDUMP)
```

SET PATHWAY Command

Use the `SET PATHWAY` command to establish attribute values for the PATHWAY object. This command configures the PATHMON process before you cold start a PATHMON environment. The attribute values cannot be changed after cold start. The Pathway attributes DUMASK and SPREBALANCEMODE were introduced in earlier release and GLINKALGORITHM is made available from L-series (L15.02) RVU.

NOTE: Among the attributes in the `SET PATHWAY` command are some that apply only to the Pathway/iTS product. Values for these Pathway-specific attributes must be included in your configuration file, even if the Pathway/iTS product is not installed at your site. This syntax diagram lists all the SET PATHWAY attributes, including those for the Pathway/iTS product. In the syntax descriptions that follow the diagram, information for each Pathway-specific attribute provides only the default value and the value you must specify if you have NonStop TS/MP but not Pathway/iTS. For full syntax descriptions for Pathway/iTS-only attributes, see the *Pathway/iTS System Management Manual*.

```
SET PATHWAY pw-attribute [ , pw-attribute ]...

pw-attribute is:
  DUMASK octalnumber
  GLINKALGORITHM link-algorithm-value
  MAXASSIGNS number
  MAXPARAMS number
  MAXSERVERCLASSES number
  MAXSERVERPROCESSES number
  MAXSTARTUPS number
  MAXTCPS number
  MAXTERMS number
  MAXDEFINES number
  MAXEXTERNALTCPS number
  MAXLINKMONS number
  MAXPATHCOMS number
  MAXPROGRAMS number
  MAXSPI number
  MAXTELLQUEUE number
  MAXTELLS number
  MAXTMFRESTARTS number
  NODEINDEPENDENT [ON | OFF]
  SCLONGPROCESSNAME [ON | OFF]
  OWNER ownerid
  SECURITY security-attribute
  SPREBALANCEMODE rbmodeval
```

pw-attribute

For information about configuration limits for environments, see **Configuration Limits and Defaults** on page 364.

`DUMASK Octalnumber`

is the default user mask (DUMASK), which is used for OSS serverclasses that do not have their UMASK attribute set. The PATHMON process uses this attribute as a file mode creation mask while starting the server process.

The `Octalnumber` entry is a three digit octal number that specifies the default maximum permissions allowed for owner, group, and others.

When this command is executed, the PATHMON process defines a value for the default UMASK attribute to configure the Pathway environment.

`GLINKALGORITHM link-algorithm-value`

specifies the value of the algorithm which PATHMON process uses to select link grant algorithm for granting link. Valid values for this attribute are REQUESTERCPU and DEFAULT. For REQUESTERCPU, link grant algorithm gives preference to static server processes running on the CPU or configured with the CPU same as that of the requester.

Table Continued

MAXASSIGNS **number**

number must be a value from 0 to 8191. The maximum number of ASSIGNs allowed by Pathway per server class and across all server classes in a given Pathway environment is 8191.

This attribute is required. It has no default.

MAXPARAMS **number**

is the maximum number of PARAM messages that you can specify across all server classes in a given Pathway environment. **number** must be a value from 0 to 4095.

This attribute is required. The default is 0.

This attribute sets the limit on the number of PARAM definitions that you can specify for Pathway server classes with the SET SERVER command. Because all PARAM messages for a server class count as one toward this limit, the number for this attribute must not exceed the value of MAXSERVERCLASSES.

MAXSERVERCLASSES **number**

is the maximum number of server class descriptions that you can add to the PATHMON configuration file.

number must be a value from 0 through 4095. To the PATHMON process to process an ALTER SERVER command, specify a maximum of 4094 for this attribute.

This attribute is required. It has no default.

MAXSERVERPROCESSES **number**

is the maximum number of server processes that you can define for all server classes using the MAXSERVERS option of the SET SERVER command.

number must be a value from 0 through 4095. To the PATHMON process to process an ALTER SERVER MAXSERVERS command, specify a maximum of 4094 for this attribute.

This attribute is required. It has no default.

The total of all MAXSERVERS values for all server classes is limited to the value of MAXSERVERPROCESSES.

MAXSTARTUPS **number**

is the maximum number of server classes that can have STARTUP messages. **number** must be a value from 0 through 4095. To the PATHMON process to process an ALTER SERVER STARTUP command, specify a maximum of 4094 for this attribute.

This attribute is required. It has no default.

Because only one SET SERVER STARTUP definition for each server class counts toward this limit, the number for this attribute must not exceed the value for MAXSERVERCLASSES.

Table Continued

MAXTCPS number

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value.

This attribute is mandatory. It has no default value.

MAXTERMS number

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

This attribute is mandatory. It has no default. If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value.

MAXDEFINES **number**

is the maximum number of DEFINE definitions that you can specify for all server classes in a Pathway environment. Each SET SERVER DEFINE attribute counts as one toward the value of this attribute.

number must be a value from 0 to 8191. The default value is 0.

MAXEXTERNALTCPS **number**

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value. The default value is 0.

MAXLINKMONS **number**

is the maximum number of ACS subsystem processes that can communicate with the PATHMON process at the same time. ACS subsystem processes manage communication between Pathway server classes and all Pathsend requestors executing within the processor of the ACS subsystem processes.

The total of the values for MAXTCPS, MAXEXTERNALTCPS, and MAXLINKMONS cannot exceed 800. A maximum of 255 TCPs, external TCPs, and ACS subsystem processes are allowed to concurrently hold links to a single server process. For other limits, see **Configuration Limits and Defaults** on page 364.

If the value for MAXLINKMONS is 0, any attempt by a ACS subsystem process to access a server controlled by the PATHMON process fails.

The default value is 0.

MAXPATHCOMS **number**

is the maximum number of PATHCOM processes that can run simultaneously within Pathway.

number must be a value from 1 through 100. The default value is 5. For other limits, see **Configuration Limits and Defaults** on page 364.

Until a Pathway environment is started, only one PATHCOM or Subsystem Programmatic Interface (SPI) process at a time can open communication with the PATHMON process.

Table Continued

MAXPROGRAMS **number**

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value.

The default value is 0.

MAXSPI **number**

is the maximum number of Subsystem Programmatic Interface (SPI) processes that can run simultaneously within the Pathway environment. SPI is part of the HPE Distributed Systems Management (DSM) facility. For more information on the SPI interface, see the *NonStop TS/MP and Pathway Management Programming Commands Manual*.

number must be a value from 1 through 100. (Until a Pathway environment is started, only one SPI process or PATHCOM process at a time can open communication with the PATHMON process.) The default value is 1. For other limits, see **Configuration Limits and Defaults** on page 364.

MAXTELLQUEUE **number**

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value.

The default value is 4.

MAXTELLS **number**

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value.

The default value is 32.

Table Continued

MAXTMFRESTARTS **number**

This attribute is specific to the Pathway/iTS product. A complete description of this attribute is given in the *Pathway/iTS System Management Manual*.

If your environment does not support Pathway/iTS, Hewlett Packard Enterprise recommends that you specify 0 for the value.

The default value is 5.

NODEINDEPENDENT [ON | OFF]

designates whether unspecified node names in Guardian file names resolve to the node name *, a generic name that designates whatever node the PATHMON process is running on after cool start.

The default value is OFF.

ON

specifies * as the default node name when the node name is left unspecified in Guardian file names defined during the current PATHCOM session and all subsequent PATHCOM sessions.

Guardian file names with unspecified node names are stored in the PATHMON configuration file using * as the node name.

OFF

s Guardian file names with unspecified node names to resolve to the node name specified by the CMDVOL command, or, if the CMDVOL is omitted, to the node of the user who started PATHCOM. For more details on the CMDVOL command, see **PATHCOM Operation Commands** on page 144.

You can explicitly designate node-independence for any PATHMON-controlled object by specifying * for the node name when you define the object. However, setting NODEINDEPENDENT to ON and then leaving out the node name when you configure an object is a simpler and more comprehensive way to accomplish node independence.

PATHCOM checks the NODEINDEPENDENT attribute when it opens the PATHMON process. If you want to take full advantage of the node-independence feature, set NODEINDEPENDENT to ON early in your PATHMON configuration session. When set, the attribute applies to all subsequent SET commands. The setting has no effect on previously specified SET commands.

Following are the file names that can be qualified by a node name when configuring a PATHMON environment. These names are designated as nodeindependent when the NODEINDEPENDENT attribute is set to ON.

Object	Associated File Name
PATHMON process	DUMP FILE file-name
Pathway object	OWNER owner-id
SERVERCLASS	ASSIGN logical-unit assign-spec GUARDIAN-LIB file name IN file-name OUT file-name OWNER owner-id PROGRAM file-name VOLUME $volume
SERVERPROCESS	GUARDIAN-SWAP $volume HOMETERM termname

Table Continued

Object	Associated File Name
TCP object*	DUMP FILE file-name GUARDIAN-LIB file-name GUARDIAN-SWAP $volume HOMETERM term-name INSPECT file-name PROGRAM file-name SWAP $volume TCLPROG file-name
TERM object*	FILE file-name INSPECT file-name PRINTER file-name TCLPROG file-name
PROGRAM object*	OWNER owner-id PRINTER file-name TCLPROG file-name
*Objects configured under Pathway/iTS only	

The NODEINDEPENDENT ON setting overrides and disables the node field of the CMDVOL command. If NODEINDEPENDENT is ON and you use CMDVOL to specify a default node other than *, you get error*2075* CMDVOL NODE VALUE CONFLICTS WITH PATHWAY NODEINDEPENDENT ON. For details, see **CMDVOL Command** on page 145.

SCLONGPROCESSNAME {ON | OFF}

This attribute enables or disables longer server process names for system generated processes. The default value is OFF.

ON

creates server processes with maximum supported length of system generated process name. (Currently 5 characters excluding “$”).

OFF

creates server processes with default length of system generated process name. (Currently 4 characters excluding “$”).

OWNER **ownerid**

is the user ID allowed to issue PATHCOM commands that directly alter the state of the PATHMON-controlled objects. The user ID must be known to the system in which PATHCOM is running. Use this attribute in conjunction with the SECURITY attribute. The value of **ownerid** is one of these:

```
[ \system-number | \*.] group-number, user-number
[ \node | \* .] group-name . user-name
```

If the NODEINDEPENDENT attribute of the 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 value is whatever node the PATHMON process is running on after cool start. The **group-name** and **user-name** are configured as you specify. For more on the * name, see the description of the SET PATHWAY NODEINDEPENDENT command, earlier in this section.

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 PATHMON’s node, and the **group-name** and **user-name** are configured as you specify.

If you omit this attribute, the node value defaults to PATHMON’s node, and the **group-name** and **user-name** values default to the user ID values of the user who started the PATHMON process. (This attribute does not use the defaults established by the CMDVOL command.)

Table Continued

`SECURITY` **`security-attribute`**

specifies the users who can issue PATHCOM commands that directly alter the state of the PATHMON-controlled objects. Before you issue the `START PATHWAY` command, the owner ID is the process-accessor ID of the PATHMON process. This security setting prevents alteration of the Pathway configuration attributes.

The security attributes are the same as the Guardian 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. The default value is “O”.	

CAUTION: If the owner of a Pathway instance is the super ID, setting the security to “N” or “A” is a security risk for your system. In TS/MP 2.6 or later versions, the default value of the SECURITY attribute of the PATHWAY configuration object is changed from “N” to “O”.

`SPREBALANCEMODE rbmodeval`

specifies the mode in which PATHMON exhibits the enhanced rebalancing of server process after processor reload. The valid values for this attribute are DISABLE, AUTO, and MANUAL. For more information about difference modes of SPREBALANCEMODE, see *HPE NonStop TS/MP 2.7 ACS Reference Manual Manual*.

Consideration

All Pathway object attributes can be changed before a cold start.

Example

These `SET PATHWAY` commands define a sample PATHMON environment:

```
SET PATHWAY DUMASK 007
SET PATHWAY GLINKALGORITHM “REQUESTERCPU”
SET PATHWAY MAXASSIGNS 15
SET PATHWAY MAXDEFINES 20
SET PATHWAY MAXLINKMONS 16
SET PATHWAY MAXPARAMS 5
SET PATHWAY MAXPATHCOMS 2
SET PATHWAY MAXSERVERCLASSES 5
SET PATHWAY MAXSERVERPROCESSES 25
```

```
SET PATHWAY MAXSPI 5
SET PATHWAY MAXSTARTUPS 5
SET PATHWAY MAXTMFRESTARTS 5
SET PATHWAY NODEINDEPENDENT ON
SET PATHWAY OWNER \*.8,8
SET PATHWAY SECURITY "U"
SET PATHWAY SPREBALANCEMODE “AUTO”
```

NOTE:

The configuration in this example does not include requestor objects that run under Pathway/iTS.

SHUTDOWN Command (Old)

The SHUTDOWN command stops the PATHMON-controlled objects.

NOTE: The SHUTDOWN2 command replaces the SHUTDOWN command. Hewlett Packard Enterprise recommends that you use the SHUTDOWN2 command for a faster, more reliable means of stopping your Pathway environment. The SHUTDOWN command is still available to maintain backward compatibility.

When you issue the SHUTDOWN command, the PATHMON process:

- Initiates the FREEZE and STOP for all server classes that PATHMON controls; stops the terminals and Pathway/iTS TCPs; writes the internal configuration and directory information to the PATHMON configuration file; and stops the PATHMON process itself.

  If a server process is engaged in a dialog at the time the SHUTDOWN command is issued, the shut down process waits until the dialog has completed.

- Notifies ACS subsystem processes and Pathway/iTS external TCPs of the shutdown, which prohibits them from sending data or requests to the server classes controlled by the PATHMON process.

The SHUTDOWN command does not stop PATHCOM.

```
SHUTDOWN [ ! ] [ , WAIT ]
```

!

directs the PATHMON process to stop all terminals, including those in the SUSPEND-RESUME and the SUSPEND-ABORT states, before stopping the Pathway/iTS TCPs, notifying the ACS subsystem processes and Pathway/iTS external TCPs, and stopping itself. If you omit this parameter, an ABORT TERM command must precede the SHUTDOWN command to stop those terminals that are not in the correct state. This parameter does not apply if the PATHMON process controls only server classes.

WAIT

directs the PATHMON process to retry this command if a terminal stop, TCP stop, or request from a process outside Pathway is rejected with a REQUEST-PENDING error. The Break key aborts this command. If you omit this attribute, this command aborts on any error.

Examples

This command stops all terminals, except those in a SUSPEND state, before stopping all server processes, Pathway/iTS TCPs, and the PATHMON process and notifying the ACS subsystem processes:

```
SHUTDOWN
```

This command stops all PATHMON-controlled objects, including terminals in a SUSPEND state:

```
SHUTDOWN!,WAIT
```

The PATHMON process retries the `SHUTDOWN!,WAIT` command if a terminal, Pathway/iTS TCP, or other process, such as a ACS subsystem process, has a request pending.

SHUTDOWN2 Command

The `SHUTDOWN2` command stops the PATHMON-controlled objects. Use the `SHUTDOWN2` command to stop the Pathway environment in one of three ways. The three ways are progressively more forceful.

The `SHUTDOWN2` command cannot be reversed, and when you have initiated the shutdown process, you cannot stop that process.

NOTE: In previous releases of Pathway, the only shutdown command available was the `SHUTDOWN` command. Hewlett Packard Enterprise recommends, however, that you use the `SHUTDOWN2` command for a faster, more reliable shutdown operation. For compatibility reasons, the `SHUTDOWN` command is still supported.

```
SHUTDOWN2 [, MODE { OR[DERLY] | AB[ORT] | IM[MEDIATE] } ]
          [, STATUS { QU[IET] | AG[GREGATE] } ]
          [, UNTIL { DONE }
          { TIMEOUT number { HRS | MINS | SECS } } ]
```

MODE

specifies the amount of force to be used when shutting down the system. The default is ORDERLY.

OR or ORDERLY

s work in progress to complete before stopping. For example, a terminal that is waiting for the completion of an I/O to a server is not stopped until the I/O completes. The PATHCOM process is suspended until the shutdown request completes or until the Break key is pressed. After shutdown, the PATHCOM prompt is returned.

AB

or

ABORT

aborts all terminals and allows all other objects such as TCPs and server classes to complete outstanding work before stopping.

The PATHCOM process is suspended until the shutdown request completes or until the Break key is pressed. After shutdown, the PATHCOM prompt is returned.

IM

or

IMMEDIATE

stops all server processes and Pathway/iTS TCPs immediately.

The PATHCOM is suspended until the shutdown request completes. After shutdown, the PATHCOM prompt is returned.

STATUS

specifies whether status information appears during a shutdown. The default is QUIET.

QU

or

QUIET

causes PATHCOM not to display any messages until the shutdown request completes. After the shutdown request is completed, PATHCOM displays the results of the shutdown.

AG

or

AGGREGATE

causes PATHCOM to display the Pathway environment status every 20 seconds while shutdown is executing. Regular status display allows the operator to monitor the shutdown process. Status continues to appear until shutdown completes.

UNTIL

specifies the time interval in which the shutdown process is to complete.

The default is DONE.

DONE

causes PATHCOM to suspend operation until shutdown completes or until the Break key is pressed.

TIMEOUT

returns control to the PATHCOM process in the specified time. The UNTIL TIMEOUT option is equivalent to pressing the Break key after the specified time.

The time is specified by a number followed by HRS, MINS, or SECS. If shutdown completes before the timeout occurs, the PATHCOM prompt is returned. If the timeout occurs and shutdown has not completed, the PATHCOM prompt returns without affecting the execution of the shutdown process.

Using the Break Key During Shutdown

You can use the Break key during shutdown to return the PATHCOM prompt without affecting the execution of the shutdown process. Returning the PATHCOM prompt allows you to investigate the state of the system (for example, issue a STATUS PATHWAY command), examine or stop any object, or escalate the shutdown process.

Escalating the Shutdown Process

After you have issued a SHUTDOWN2 command with MODE ORDERLY or MODE ABORT, you can escalate the shutdown process by pressing the **Break** key and entering a more forceful SHUTDOWN2 option.

Considerations

- The SHUTDOWN2 command performs:

  Stops all Pathway/iTS TERM objects Stops all Pathway/iTS TCPs Stops all SERVER objects Stops the PATHWAY object

- These PATHCOM commands are disabled when the PATHMON process is shutting down the PATHMON environment:

Command Type	Commands Disabled During SHUTDOWN State	
Control commands	PRIMARY PATHMON	START PATHWAY
	SWITCH PATHMON	
Server class commands	ADDALTERDELETE	STARTTHAW
TCP commands(Pathway/iTS product only)	ADD ALTERDELETEPRIMARY	REFRESH-CODESTARTSWITCH
Program commands(Pathway/iTS product only)	ADDALTER	DELETERUN
Terminal commands(Pathway/iTS product only)	ADDALTERDELETEINSPECT	RESUMESTARTSUSPEND
Tell commands(Pathway/iTS product only)	DELETE	TELL

- Use the ORDERLY option, for example, for epilogue processing (when the application requires the server process to write out totals before stopping).
- The ABORT option allows send operations to complete, but work does not necessarily stop on transaction boundaries. If it is important to your application to complete outstanding transactions, use

the ORDERLY option. The ABORT option, like the ORDERLY option, allows epilogue processing to complete before the server process is stopped.

- Upon successful completion of the shutdown process, the PATHMON process is stopped.
- For more information on strategies for shutting down a Pathway environment, see **Starting and Stopping a PATHMON Environment** on page 33.

Examples

This command causes PATHCOM to stop all processes controlled by the PATHMON process. The PATHCOM process is suspended until the shutdown request completes:

```
SHUTDOWN2, MODE IMMEDIATE
```

This command causes PATHCOM to display the Pathway environment status every 20 seconds during an orderly shutdown:

```
SHUTDOWN2, STATUS AGGREGATE
```

This command returns control to the PATHCOM process in 30 seconds or earlier if the shutdown completes, or, if the shutdown has not completed, after 30 seconds have passed:

```
SHUTDOWN2, UNTIL TIMEOUT 30 SECS
```

START PATHWAY Command

Use the START PATHWAY command to start a Pathway environment.

```
START PATHWAY { COOL } [!]
              { COLD } [!]
```

COOL

starts the PATHMON-controlled objects using the configuration information contained in an existing PATHMON configuration file. All PATHWAY, PATHMON, and SERVER object definitions, as well as Pathway/iTS TCP, TERM, and PROGRAM object definitions, remain in the file. The PATHMON process restores the names of the log files that were in use when the Pathway environment was last running.

If the PATHMON process name is different than the PATHMON name saved in the PATHMON configuration file, the PATHCOM process notifies the operator. The operator can either continue the START PATHWAY COOL command or abort it.

COLD

starts the PATHMON-controlled objects using the configuration information specified with the SET PATHWAY commands. All SERVER object definitions, and Pathway/iTS TCP, TERM, and PROGRAM object definitions, must be specified using the appropriate SET command following this command.

If a configuration file already exists, the PATHCOM process notifies the operator. The operator can either continue the START PATHWAY COLD command or abort it in preparation for cool starting the environment.

!

specifies that the command is to be run without any warnings or prompts.

When used with COLD, the PATHMON process writes over the existing PATHMON configuration file with the current configuration information.

When used with COOL, the PATHMON process uses the existing PATHMON configuration file to restart Pathway and does not issue a warning if the PATHMON process name is different from the PATHMON name saved in the PATHMON configuration file.

Considerations

- You must define values for the required PATHMON and PATHWAY object attributes before using the START PATHWAY command with the COLD option.
- You can issue a START PATHWAY command for a different (remote) Pathway environment (one controlled by a PATHMON process different from the PATHMON process controlling your local Pathway environment).

Examples

This command starts a Pathway environment and causes the PATHMON process to allocate a new PATHMON configuration file:

```
START PATHWAY COLD!
```

This command starts a Pathway environment but causes the PATHMON process to use the existing PATHMON configuration file:

```
START PATHWAY COOL
```

STATUS LINKMON Command

Use the STATUS LINKMON command to display information about link manager (LINKMON/ACS subsystem) processes that have links to server processes.

```
STATUS [ / OUT list-file / ]
   { [ LINKMON ] L\node.$process-name }
   { LINKMON * }
```

`OUT list-file`

directs output to the file that you specify; this can 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.

`L\node.$process-name`

specifies the name of the link manager (LINKMON/ACS subsystem) process.

`L`	identifies the process as a ACS subsystem processes.
`\node`	specifies the name of the NonStop system on which the ACS subsystem processes are running.
`$process-name`	specifies the name of the LINKMON processes or ACS subsystem processes. LINKMON process names are in the form $ZLnn, where nn is the number of the processor where the LINKMON process resides. ACS subsystem process name is in the form $ZPnn, where nn is the number of the processor where the ACS subsystem PB process resides.

`LINKMON *`

displays the status of all LINKMON processes or ACS subsystem Process Broker processes currently known to the PATHMON process.

Display Format

The format of the display returned by the `STATUS LINKMON` command is shown in **STATUS LINKMON Display Format** on page 185. This format is similar to the one for the `STATUS TCP` command, except that the `DETAIL` option is not available with the `STATUS LINKMON` command. The fields are described in the following section:

STATUS LINKMON Display Format

LINKMON	STATE	ERROR	INFO	PROCESS	CPU
lm-name	state	pw-error	info	process-name	cpu
. . .	...	. ..	. . .	. ..	. . .

The fields in this display are as follows:

Display Field	Description
LINKMON	Full name of the LINKMON process, including the node name
STATE	Current state of the LINKMON process. The only possible value is:

Table Continued

Display Field	Description	
	RUNNING	The LINKMON process is running and communicating with the PATHMON process
ERROR	Pathway error number	
INFO	Additional information regarding the error number. See the error messages listed in **PATHMON Messages (Numbers 1000-1499)** on page 245 through **LINKMON Log Messages** on page 306.	
PROCESS	Process name of the LINKMON process	
CPU	processor number in which the LINKMON process is currently running	

Examples

This command displays status information for all LINKMON processes communicating with the PATHMON process:

```
STATUS LINKMON *
```

These commands display status information for a specific LINKMON process running on a specific node:

```
STATUS LINKMON L\NWREG.$ZL05
STATUS L\NWREG.$ZL05
```

This command directs status information to a specified output file:

```
STATUS/OUT STATFLE/LINKMON L\CPTNO.$ZL05
```

STATUS PATHMON Command

Use the `STATUS PATHMON` command to display the status of the PATHMON process, the PATHMON configuration file, and the logging files. This command also displays a list of the processes currently communicating with the PATHMON process.

```
STATUS [ / OUT list-file / ] PATHMON
```

`OUT` **`list-file`**

directs output to the file that you specify; this can 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.

Display Format

The format of the display returned by the `STATUS PATHMON` command is shown in **STATUS PATHMON Display Format** on page 186. The fields are described in the following section:

STATUS PATHMON Display Format

```
PATHMON \node.$process-name-- STATE=pmon-state CPUS pri : backup
  PATHCTL (file-state) $pathctl-file-name   [ ERROR = num ]
  LOG1 [S] [E] (file-state) $log1-file-name [ ERROR = num ]
```

```
  LOG2 [S] [E] (file-state) $log2-file-name [ ERROR = num ]

    REQNUM   FILE   PID   PAID WAIT
[A] reqnum   type   pid   paid wait-cause
      .       .      .      .     .
      .       .      .      .     .
      .       .      .      .     .
```

The fields in this display are as follows:

Display Field	Description	
PATHMON - STATE	PATHMON process name and current state. Possible state values are: STARTING A cold or cool start has not completed. RUNNING A cold or cool start has completed. SHUTTINGDOWN A shutdown request has not completed.	
CPUS	Primary and backup CPUs in which the PATHMON process is running. The processor numbers displayed are the current CPUs for these processes and not necessarily the CPUs configured for the PATHMON process. Use the `INFO PATHMON` command to find the configured backup processor number. If the PATHMON backup process is not running, the PATHMON process does not display a number for the backup process.	
PATHCTL	The current state and name of the file. Possible state values are: CLOSED The file is closed. OPEN The file is open.	
LOG1 and LOG2	The type of logging defined, current state, and name of the indicated log file. Possible logging types are: S The log file is open for status logging. E The log file is open for event logging. Possible state values are: CLOSED The file is closed. OPEN The file is open.	
ERROR	Last error for the indicated file	
REQNUM	The PATHMON process internal identifier of a requestor. An A next to the requestor number indicates that the PATHMON process is completing a request for a requestor that aborted.	
FILE	Requestor type. Possible values are:	
	LINKMON	The requestor is a LINKMON process.

Table Continued

Display Field	Description	
	LOG	The task that writes log messages to the log file. This is always **reqnum** = 0 and is displayed only if the task is waiting.
	PATHCOM	The requestor is a PATHCOM process.
	TCP	The requestor is a TCP running in this Pathway environment.
	EXT TCP	The requestor is a TCP running in another Pathway environment.
	SYS	The requestor is the operating system; that is, a system message is being processed.
	SPI	The requestor type is SPI.
	For information about TCP and EXT TCP types, see the *Pathway/iTS System Management Manual*.	
PID	Process ID of the requestor process (PATHMON, TCP, or LINKMON process)	
PAID	Process-accessor ID of the requestor process	
WAIT	The reason a request is waiting. Possible values are:	
	IO	The request is waiting for an I/O operation to complete.
	LOCK	The request is waiting for an object locked by another requestor.
	LOCK-QUEUE	The requestor is waiting for space in the internal lock queue.
	BUF	The request is waiting for internal buffer space to become available.
	LOG-QUEUE	The request is waiting for an entry in the internal status logging message queue to become available.
	TIMEOUT	The request is waiting for a period of time to elapse before retrying an operation.
	PROG-DONE	The request is waiting for a RUN PROGRAM to finish.
	NEWPROCESS	The request is waiting for a NEWPROCESS NOWAIT operation to complete.

The WAIT field displays file names associated with the reason a request is waiting and a list of the locks owned by each requestor. The list is labeled with the word LOCKED.

Example

This command requests that the PATHMON status display be written to the file named STATFLE:

```
STATUS/OUT statfle/PATHMON
```

STATUS PATHWAY Command

Use the STATUS PATHWAY command to display the number of PATHMON-controlled objects that are in each possible state. This command provides an overview of the Pathway environment.

You can also use the STATUS PATHWAY command with the SHUTDOWN2 command to monitor the progress of a Pathway environment shutdown.

```
STATUS [ / OUT list-file /] PATHWAY , option [ , option] ...
option is:
          COUNT number
          INTERVAL number { HRS | MINS | SECS }
```

OUT list-file

directs output to the file that you specify; this can 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.

COUNT number

determines the number of times that the status screen appears until the Break key is used, or until the Pathway environment is shut down, whichever occurs first. The default count is 1.

INTERVAL number HRS | MINS | SECS

determines the delay between the status displays. The delay is based on the value specified by **number** in hours, minutes, or seconds. The default is no delay between STATUS PATHWAY displays.

The STATUS PATHWAY display formats are described in this discussion. For information about the Pathway/iTS objects that appear in this display (EXTERNAL TCPS, TCPS, and TERMS), see the *Pathway/iTS System Management Manual*.

Display Format—Running

STATUS PATHWAY Display Format (Running) on page 189 shows the format of the display returned by the STATUS PATHWAY command while the system is running.

STATUS PATHWAY Display Format (Running)

```
PATHWAY -- STATE = RUNNING

                 RUNNING

EXTERNALTCPS     number

LINKMONS         number

PATHCOMS         number

SPI              number

                                                              FREEZE

                 RUNNING     STOPPED     THAWED      FROZEN      PENDING
```

Table Continued

```
                  number      number      number      number      number

                  RUNNING     STOPPED     PENDING
SERVERPROCESSES   number      number      number

                  RUNNING     STOPPED     PENDING
TCPS              number      number      number

                  RUNNING     STOPPED     PENDING     SUSPENDED
TERMS             number      number      number      number
```

Display Format—During Shutdown

STATUS PATHWAY Display Format (After Shutdown Failure) on page 190 shows the format of the display returned by the `STATUS PATHWAY` command during a shutdown.

STATUS PATHWAY Display Format (After Shutdown Failure)

```
PATHWAY --STATE = SHUTTING-DOWN (IN PROGRESS) - STARTED 2:02:39
                                                        STOPPED 4:08:44

             RUNNING

EXTERNALTCP  number
S            number
LINKMONS     number
PATHCOMS     number
SPI


                                                                FREEZE
             RUNNING     STOPPED     THAWED      FROZEN      PENDING
SERVERCLASS  number      number      number      number      number
ES


             RUNNING     STOPPED     PENDING
SERVERPROCE  number      number      number
SSES


             RUNNING     STOPPED     PENDING

TCPS         number      number      number
```

Table Continued

```
             RUNNING      STOPPED      PENDING      SUSPENDED
TERMS        number       number       number       number
```

Display Format—Shutdown Failure

STATUS PATHWAY Display Format (After Shutdown Failure) on page 191 shows the format of the display returned by the `STATUS PATHWAY` command when a shutdown has failed.

STATUS PATHWAY Display Format (After Shutdown Failure)

```
PATHWAY --STATE = SHUTTING-DOWN (FAILED) - STARTED 4:01:39
                                                   STOPPED 4:08:44

             RUNNING

EXTERNALTCP  number
S            number
LINKMONS     number
PATHCOMS     number
SPI



                                                              FREEZE
             RUNNING      STOPPED      THAWED       FROZEN    PENDING
SERVERCLASS  number       number       number       number    number
ES



             RUNNING      STOPPED      PENDING
SERVERPROCE  number       number       number
SSES



             RUNNING      STOPPED      PENDING
TCPS         number       number       number



             RUNNING      STOPPED      PENDING      SUSPENDED
TERMS        number       number       number       number
```

Calculating Totals

Totals for each object can be calculated from these displays as follows:

Total…	Is Calculated By Adding Together the Fields…
Total server classes	RUNNING + STOPPED OR THAWED + FROZEN+ FREEZE PENDING
Total server processes	RUNNING + STOPPED + PENDING
Total TCPs (Pathway/iTS)	RUNNING + STOPPED + PENDING
Total terminals (Pathway/iTS)	RUNNING + STOPPED + PENDING + SUSPENDED

Example

This command writes the status display to a file named STATFLE:

```
STATUS/OUT STATFLE/PATHWAY
```

STOP PATHMON Command

Use the `STOP PATHMON` command to stop the PATHMON process pair. To successfully run this command, you must first issue a `SHUTDOWN2 PATHWAY` command.

```
STOP PATHMON
```

Consideration

All processes started by the PATHMON process and still running when the `STOP PATHMON` command completes successfully are orphaned.

To avoid orphaned processes, first use the `STATUS PATHWAY` command, before executing the `STOP PATHMON` command, to determine what type of processes are running. Use the STATUS command on each type of object to find the names of the active processes, then stop each process. After all processes are stopped, then run the `STOP PATHMON` command.

Errors

This table lists the most common error that can occur during the processing of the `STOP PATHMON` command:

This Message…	Is Displayed When…
1045 REQUEST NOT VALID FOR CURRENT STATE	You have issued a `STOP PATHMON` command but the PATHMON process is not in the shutting-down state.
	Issue a `SHUTDOWN2 PATHWAY` command before issuing the `STOP PATHMON` command.

SWITCH PATHMON Command

Use the `SWITCH` command to direct the PATHMON process to exchange the function of its primary process with the function of its backup process while the PATHMON process is running.

This command does not change the PATHMON processor definitions in the PATHMON configuration file.

```
SWITCH PATHMON
```

Considerations

- Use the `STATUS PATHMON` command to identify the CPUs in which the PATHMON primary and backup processes are actually running after you have repeatedly entered the `SWITCH` command.
- Whenever the PATHMON primary or backup process is stopped more than 10 times by a `STOP` command, the PATHMON process delays creating a backup process. The PATHMON process delays creating a backup process by 1 minute if stopped 11 times, 2 minutes if stopped 12 times, and so forth.

  However, the PATHMON process does not delay creating a backup process if any of these occur:

  - The processor in which the PATHMON primary or backup process is executing fails.
  - A `PATHCOM SWITCH` command is run.
  - A `CONTROL PATHMON BACKUPCPU` command is run.

  When the PATHMON process does not delay the creation of a backup process, it resets its internal counter. Consequently, creation of a backup process has to fail ten more times before the PATHMON process delays creating the backup process.

- After successfully executing a `SWITCH PATHMON` command, the PATHMON process logs this status message:

  ```
  STATUS - *1077* TAKEOVER BY BACKUP
  ```

SERVER Commands

This section describes the PATHCOM commands that define and control SERVER objects; the commands are listed in alphabetic order. The server class commands are as follows:

To Perform these Tasks…	Use These PATHCOM Commands…
Define and remove server classes	`SET SERVERADD SERVERDELETE SERVER`
Control servers in server class	`START SERVER STOP SERVER`
Change server class attributes	`ALTER SERVERRESET SERVER`
Control requests to servers	`FREEZE SERVERTHAW SERVER`
Obtain information about	`INFO SERVERSHOW SERVERSTATS SERVERSTATUS SERVER`

With these commands you can:

- Set and modify server class attributes
- Add and delete server classes
- Start and stop server processes
- Display server attribute and status information
- Control server communication

NOTE: Information in this section about the interaction between TCPs and server attributes applies to your environment only if you are using the Pathway/iTS product.

ADD SERVER Command

Use the `ADD SERVER` command to enter the name and description of a server class into the PATHMON configuration file. Use this command after defining a server class with the `SET SERVER` command.

```
ADD SERVER server-class [ , server-attribute ]...
```

`server-class`

specifies the name of a server class. A SERVER object name can be from 1 to 15 alphanumeric or hyphen characters and must start with a letter, be unique within the Pathway environment, and not be a Pathway reserved word.

`server-attribute`

specifies an attribute describing a server class. A server class attribute consists of a keyword and a value. The value you enter overrides the value previously established with the `SET SERVER` command. Use any of the attributes listed for the `SET SERVER` command. All attributes specified must be consistent with the process type defined for the server class.

Consideration

The ADD SERVER command does not affect values established with the SET SERVER or RESET SERVER commands. Use the RESET SERVER command to change all of the values to their defaults, or use the SET SERVER command to replace only specific values.

Examples

This command adds a definition for the server class CLASS-1 to the PATHMON configuration file:

```
ADD SERVER CLASS-1
```

This command adds the server class CLASS-2 to the PATHMON configuration file and defines its AUTORESTART and DEBUG attributes:

```
ADD SERVER CLASS-2, AUTORESTART 4, DEBUG ON
```

ALTER SERVER Command

Use the ALTER SERVER command to change the attribute values of a server class that was previously added to the PATHMON configuration file. See the SET SERVER command for a list of the server class attributes.

All server processes in the server class must be stopped before you issue this command.

```
ALTER [ SERVER ] server-class
   { , server-attribute                              }
   { , DELETE delete-option                          }
   { , RESET server-keyword                          }
   { , RESET (server-keyword [ , server-keyword ]...) }

delete-option is:

   [ ASSIGN logical-unit  ]
   [ DEFINE define-name   ]
   [ PARAM parameter-name ]
   [ PROCESS process-name ]
   [ STARTUP              ]
```

server-class

specifies the name of a previously defined and added server class.

server-attribute

specifies an attribute describing a server class. A server class attribute consists of a keyword and a value. The value you enter overrides the value previously established with the SET SERVER command. Use any of the attributes listed for the SET SERVER command. All attributes specified must be consistent with the process type defined for the server class.

DELETE **delete-option**

clears the value for the specified attribute. This parameter only deletes one attribute value per command: for example, one ASSIGN or PARAM value.

Table Continued

RESET

changes an existing server class attribute to the default value. This parameter can reset one or all values for a given attribute, for example, all ASSIGN or PARAM values. If you set multiple values for an attribute (for example, the *ASSIGN*, *DEFINE*, and *PARAM* parameters), the RESET option deletes only the values that you specify.

If an attribute is required and no Pathway default value exists, the RESET option generates a syntax error and leaves the current attribute value unchanged.

server-keyword

specifies a single server class attribute keyword or several keywords separated by commas and enclosed in parentheses. The RESET option changes the values for these SET SERVER command attributes:

ARGLIST	GUARDIAN-SWAP	PRI
ASSIGN	HIGHPIN	PROCESS
(ASSIGN logicalunit)	HOMETERM	PROGRAM
AUTORESTART	IN	SECURITY
CPUS	LINKALGORITHM	STARTUP
CREATEDELAY	LINKDEPTH	STDERR
CWD	MAXLINKS	STDIN
DEBUG	MAXSERVERS	STDOUT
DEFINE	NUMSTATIC	TMF
(DEFINE definename)	OUT	UMASK
DELETEDELAY	OWNER	VOLUME
ENV	PARAM	
GUARDIAN-LIB	(PARAM paramname)	

Consideration

For information on altering associative servers, see the discussion of associative servers in **Maintaining a PATHMON Environment** on page 81.

Errors

This table lists the most common error that can occur during the processing of the ALTER SERVER command:

This Message…	Is Displayed When…
1102 TOO MANY SERVER ENTRIES	You have tried to alter a server class configuration but you have already added the maximum number of SERVER object entries (that is, 4095) supported by the PATHMON process.
	The command failed because the PATHMON process runs the `ALTER SERVER` command by creating a new server class entry with the altered values before deleting the existing server class entry. When the maximum number of server class entries already exists, the PATHMON process does not have the space to create a new server class.
	To avoid this problem, define and add the maximum number of SERVER entries allowed minus 1 (that is, 4094 SERVERs).

Examples

These commands change various attribute values for the specified server classes:

```
ALTER SERVER CLASS-1, PRI 10
ALTER CLASS-2, RESET (TMF, MAXLINKS)
ALTER SERVER CLASS-3, DELETE PARAM SWITCH-1
ALTER SERVER CLASS-4, DEFINE =EMP, CLASS MAP, FILE
$D.APPL.EMP
```

These commands produce the same result:

```
ALTER SERVER CLASS-1, RESET PROCESS $ABC
ALTER SERVER CLASS-1, DELETE PROCESS $ABC
```

DELETE SERVER Command

Use the `DELETE SERVER` command to remove a server class description from the PATHMON configuration file. All servers in the server class must be stopped before the PATHMON process can delete a SERVER object.

```
DELETE [SERVER] { server-class }
{ ( server-class [ , server-class ]... ) }
```

Column Head

`server-class`

specifies the name of a SERVER object.

Examples

These commands delete the specified server classes:

```
DELETE SERVER CLASS-1
DELETE (CLASS-2, CLASS-3)
DELETE CLASS-4
```

FREEZE SERVER Command

Use the `FREEZE SERVER` command to prohibit all link managers, such as ACS subsystem processes, or TCPs, from sending requests to a server class.

If a server process has incomplete requests (requests for which links have not yet been granted) or outstanding requests (such as an active dialog), the PATHMON process cannot freeze the server class until after these requests complete. While waiting for the requests to complete, the server is in the FREEZE-PENDING state, and no new requests can be initiated.

```
FREEZE { [ SERVER ] server-class                                    }
       { [ SERVER ] (server-class [, server-class ]...) }
       { SERVER *                                                        }
   [ !       ]
   [ , WAIT ]
```

server-class

specifies the name of a previously defined and added server class.

`SERVER *`

prohibits all link managers, such as ACS subsystem processes, or TCPs, known to the PATHMON process from sending requests to a server class.

`!`

prohibits all link managers from sending requests to a server class; the SCREEN COBOL STOP-MODE special register value is ignored.

`WAIT`

directs the PATHMON process to complete this command pending acceptance by the link managers of the communication freeze on the server class. This option allows you to monitor the freeze operation as it takes place. The PATHMON process attempts to freeze each server class and reports on each one as frozen or pending freeze. The process then loops until all server classes are frozen.

You can abort the `WAIT` option by pressing the **Break** key.

Considerations

- If you are using Pathway/iTS, consider these:
  - Unless ! is specified, the freeze does not take effect until all SCREEN COBOL STOP-MODE special registers have a value of zero and no server in the server class has incomplete or outstanding requests. In the interim, the server class is put into the FREEZE-PENDING state.
  - If a SCREEN COBOL requestor attempts to send a request to a frozen server class and the SEND statement is coded without an ON ERROR clause, these occurs:

- If the terminal is not in TMF transaction mode, the terminal pauses until the server class is thawed.
- If the terminal is in TMF transaction mode, the terminal pauses and the TMF subsystem backs out the current transaction. After the `THAW` command resumes terminal activity, the TCP directs the TMF software to restart the transaction at the BEGIN-TRANSACTION verb and to assign a new transaction identifier, and increments the special register RESTARTCOUNTER by 1.

If the SEND statement is coded with an ON ERROR clause, the TCP passes control to the screen program for the appropriate action.

- Unlike a TCP, the ACS subsystem processes do not call any TMF procedures or increment any special TMF registers. It is the responsibility of the Pathsend process to detect any errors (by an error code) and proceed as required by the application.

Examples

These commands freeze the specified server classes:

```
FREEZE SERVER CLASS-1
FREEZE (CLASS-2,CLASS-3) !
```

This command immediately freezes all the server classes in the Pathway environment:

```
FREEZE SERVER * !
```

This command freezes all the server classes in the system and waits until all terminals are frozen to complete the command:

```
FREEZE SERVER *, WAIT
```

INFO SERVER Command

Use the `INFO SERVER` command to display the attribute values defined in the PATHMON configuration file for a server class.

```
INFO [ / OUT list-file / ]
    { [ SERVER ] server-class                              }
    { [ SERVER ] ( server-class [ , server-class ] ...) }
    { SERVER *                                             }

    [ , OBEYFORM ]
```

`OUT` **`list-file`**

directs output to the named list file; this can 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.

`server-class`

specifies the name of a previously defined and added SERVER object.

Table Continued

SERVER *

displays attribute values of all server classes in the PATHMON configuration file. This command does not display information about servers controlled by a PATHMON process in a different PATHMON environment.

OBEYFORM

displays the information in the format you would use to set up a PATHMON configuration file; each attribute appears as a syntactically correct SET command. PATHCOM adds a RESET SERVER command before and an ADD SERVER command after each SERVER object description.

Display Format

INFO SERVER displays both the server class attribute values you explicitly defined and default values for the attributes you did not explicitly define. The attributes are displayed in alphabetic order after PROCESSTYPE.

These attributes are displayed for all server classes:

PROCESSTYPE	HOMETERM	PRI
AUTORESTART	LINKALGORITHM	PROGRAM
CREATEDELAY	LINKDEPTH	SECURITY
DEBUG	MAXSERVERS	TMF
DELETEDELAY	NUMSTATIC	UMASK
HIGHPIN	OWNER	VOLUME

If PROCESSTYPE is GUARDIAN, these attributes are displayed only if they have values defined:

ASSIGN	IN	PROCESS
DEFINE	OUT	STARTUP
GUARDIAN-LIB	PARAM	

If PROCESSTYPE is OSS, these attributes are displayed only if they have values defined:

ARGLIST	ENV	STDERR
CWD	GUARDIAN-LIB	STDIN
DEFINE	PROCESS	STDOUT

The CWD attribute appears when a value is set either by the SET SERVER CWD command or by the CMDCWD command. All OSS pathnames, argument lists, and environment variables are displayed as quoted strings, regardless of whether there are embedded quotes (similar to the way PARAM values are displayed).

Considerations

- This command returns information about a server class only after the description is 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 configuration. You must add the START PATHWAY command at the end of the OUT file before using the file to start the PATHWAY object.

Examples

These command requests information for two server classes:

```
INFO (CLASS-2,CLASS-3)
```

This command:

```
INFO SERVER CLASS-1, OBEYFORM
```

returns the following screen display:

```
RESET SERVER
SET SERVER PROCESSTYPE GUARDIAN
SET SERVER AUTORESTART 0
SET SERVER CPUS (2:1,3:2,0:1)
SET SERVER CREATEDELAY 1 MINS
SET SERVER DEBUG OFF
SET SERVER DEFINE =EMP, CLASS MAP, FILE \SYS.$D.APPL.EMP
SET SERVER DELETEDELAY 10 MINS
SET SERVER HIGHPIN OFF
SET SERVER HOMETERM \*.$TERM
SET SERVER LINKALGORITHM "REQUESTERCPU"
SET SERVER LINKDEPTH 1
SET SERVER MAXSERVERS 5
SET SERVER NUMSTATIC 2
SET SERVER OWNER \TS.8,8
SET SERVER PRI 134
SET SERVER PROGRAM \*.$BANK1.TESTING.POBJ
SET SERVER SECURITY "N"
SET SERVER TMF ON
SET SERVER VOLUME \*.$BANK1.TESTING
ADD SERVER CLASS-1
```

This command:

```
INFO SERVER CLASS-1
```

returns the following screen display:

```
SERVER CLASS-1
  PROCESSTYPE GUARDIAN
  AUTORESTART 0
  CPUS (2:1,3:2,0:1)
  CREATEDELAY 1 MINS
  DEBUG OFF
  DEFINE =EMP,CLASS MAP,FILE \SYS.$D.APPL.EMP
  DELETEDELAY 10 MINS
  HIGHPIN OFF
  HOMETERM \*.$TERM
  LINKALGORITHM “REQUESTERCPU”
  LINKDEPTH 1
```

```
MAXSERVERS 5
NUMSTATIC 2
OWNER \TS.8,8
PRI 134
PROGRAM \*.$BANK1.TESTING.POBJ
SECURITY "N"
TMF ON
VOLUME \*.$BANK1.TESTING
```

In the example, the generic node name * specified for the HOMETERM, PROGRAM, and VOLUME attribute values indicates that node name for these attributes will default to the node where the PATHMON process is currently running.

NOTE: The SERVER OWNER and SERVER SECURITY attributes serve as security checks for a Pathsend process attempting to access a Pathway server class. TCPs do not use these attributes.

This command directs the information to a particular file:

```
INFO/OUT SERVFLE/SERVER *, OBEYFORM
```

This sequence of commands captures a configuration for a complete PATHMON environment (minus the `START PATHWAY` command) in the command file called NEWCONFG. (Notice that this configuration includes Pathway/iTS objects: the TCP, TERM, and PROGRAM objects).

```
=INFO/OUT NEWCONFG/PATHMON, OBEYFORM
=INFO/OUT NEWCONFG/PATHWAY, OBEYFORM
=INFO/OUT NEWCONFG/TCP *, OBEYFORM
=INFO/OUT NEWCONFG/TERM *, OBEYFORM
=INFO/OUT NEWCONFG/SERVER *, OBEYFORM
=INFO/OUT NEWCONFG/PROGRAM *, OBEYFORM
```

RESET SERVER Command

Use the `RESET SERVER` command to change the values for server class attributes from the ones you defined with the `SET SERVER` command to the default values. This command does not change the attributes of a server class already added to the PATHMON configuration file. This command is enhanced to reset the new server attribute UMASK. This attribute is not a mandatory parameter. However, if this attribute is set in a PATHCOM session, then RESET SERVER command will reset this attribute and will not display the same if `SHOW SERVER` command is issued, see Example 2 below.

```
RESET SERVER [ server-keyword [ , server-keyword ]... ]

server-keyword is:
ARGLIST     GUARDIAN-SWAP PROCESS
ASSIGN      HOMETERM      PROCESSTYPE
AUTORESTART     IN            PROGRAM
CPUS        LINKALGORITHM SECURITY
CREATEDELAY LINKDEPTH     STARTUP
CWD         MAXLINKS      STDERR
DEBUG       MAXSERVERS        STDIN
DEFINE      NUMSTATIC     STDOUT
DELETEDELAY OUT           TMF
ENV         OWNER         UMASK
            PARAM         VOLUME
```

server-keyword

specifies one or more attributes to be reset. For attributes have more than one value, such as ENV and PARAM, specifies one or more, or all values to be reset.

If you omit **server-keyword**, this command resets all attribute values to the defaults.

Considerations

- Some required server class attributes have no default values. If you include a required attribute in the RESET SERVER command, the attribute is set to a null value and must be set again before you add the server class description to the PATHMON configuration file.
- If an attribute is not required and does not have a default value, the RESET SERVER command deletes the value for the attribute. When you set multiple values for an attribute (for instance, the ASSIGN and PARAM attributes), the RESET SERVER command deletes all the values set for the attribute.

Example 1

This command resets all attribute values:

```
RESET SERVER
```

This command deletes all of the file assignments for the server class:

```
RESET SERVER ASSIGN
```

This command resets a specific DEFINE definition:

```
RESET SERVER DEFINE =employee
```

This command resets the values for the specified attributes:

```
RESET SERVER CREATEDELAY,MAXSERVERS
```

Example 2

The SHOW SERVER command before RESET SERVER displays the new attribute.

```
SERVER
PROCESSTYPE GUARDIAN
.
.
SECURITY "N"
TMF OFF
UMASK Octalnumber
VOLUME \node.$volume.subvolume
```

The SHOW SERVER command after RESET SERVER does not display the new attribute.

```
SERVER
PROCESSTYPE GUARDIAN
.
.
SECURITY "N"
TMF OFF
VOLUME \node.$volume.subvolume
```

SET SERVER Command

Use the `SET SERVER` command to define values for server class attributes. Use this command before using the `ADD SERVER` command.

```
SET SERVER server-attribute [ , server-attribute ]...

server-attribute is:
    PROCESSTYPE { GUARDIAN | OSS }
    ARGLIST argument [,...]
    ASSIGN logical-unit , assign-spec
    AUTORESTART number
    CPUS { ( primary:backup [, primary:backup ]...) |
    ( cpu [ , cpu ]... ) |
    ( cpu (cpu-wt) [ , cpu (cpu-wt) ]... )
    }
    CREATEDELAY number { HRS | MINS | SECS | CSECS }
    CWD oss-pathname
    DEBUG { ON | OFF }
    ( DEFINE define-name , define-attribute-spec
    [, define-attribute-spec ] ... )
    DELETEDELAY number { HRS | MINS | SECS }
    ENV name=value [ ,... ]
    GUARDIAN-LIB file-name
    HIGHPIN { ON | OFF }
    HOMETERM file-name
    IN file-name
    LIKE server-class
    LINKALGORITHM link-algorithm-value
    LINKDEPTH number
    MAXLINKS number
    MAXSERVERS number
    NUMSTATIC number
    OUT file-name
    OWNER ownerid
    PARAM parameter-name parameter-value [ , ... ]
    PRI priority
    PROCESS $process-name [ ( process-attribute , ... ) ]
    PROGRAM { file-name | oss-pathname }
    SECURITY security-attribute
    STARTUP string
    STDERR oss-pathname
    STDIN oss-pathname
    STDOUT oss-pathname
    TIMEOUT number { HRS | MINS | SECS }
    TMF { ON | OFF }
         UMASK Octalnumber
    VOLUME volume-spec
logical-unit is:

   { [ program-unit . ] } logical-file-name
   {       * .          }

assign-spec is:

   [ file-name                        ]
```

```
   [ [ file-name ] , create-spec ... ]

create-spec is:

   { EXT [ ( ] pri-extent-size [ ) ]                }
   { EXT ( [ pri-extent-size ] , sec-extent-size ) }
   { EXCLUSIVE                                                    }
   { SHARED                                                        }
   { PROTECTED                                                     }
   { I/O                                                           }
   { INPUT                                                        }
   { OUTPUT                                                        }
   { CODE file-code                                             }

process-attribute is:

   [ ASSOCIATIVE { ON | OFF}                              ]
   [ CPUS primary:backup | cpu ]
   [ DEBUG { ON | OFF }         ]
   [ GUARDIAN-SWAP $volume      ]
   [ HOMETERM file-name         ]
   [ PRI number                 ]
```

server-attribute

Specifies one or more attributes of the SERVER object being defined. The PROGRAM attribute is the only required attribute.

These attributes describe Guardian server processes only:

```
ASSIGN PARAM
IN     STARTUP
OUT
```

These attributes describe OSS server processes only:

```
ARGLIST STDERR
CWD     STDIN
ENV     STDOUT
```

All other SERVER object attributes describe both Guardian and OSS server processes. For information about configuration limits, also see **Configuration Limits and Defaults** on page 364.

`PROCESSTYPE GUARDIAN | OSS`

specifies the type of servers in the server class.

`GUARDIAN`

All servers in this server class are Guardian processes.

`OSS`

All servers in this server class are OSS processes.

If you omit this attribute, the default is GUARDIAN.

All attributes specified for a server class must be consistent with the type specified in this attribute. Therefore, Hewlett Packard Enterprise recommends that you set this attribute first, before setting other attributes.

This attribute is required for all OSS server processes.

Table Continued

ARGLIST **argument**

specifies a Open System Services (OSS) startup argument list, which is a list of strings separated by commas that is made available to OSS server processes in the argv[] array.

You can specify from 2 to 24,000 characters for the ARGLIST attribute; a null string is valid. The length of the specified string is calculated as the number of bytes of storage needed. The length after quotes is removed and a null byte is added for each argument. However, the combined total (storage) length of the values set for ARGLIST and ENV must not exceed 24,000 characters.

The program name that is automatically passed as the first argument (argv[0]) does not need to be specified by the user, nor is it included in the 24,000 character maximum.

If you omit this attribute, the default is an empty argument list.

△ **CAUTION:** Neither PATHCOM nor the PATHMON process check for escape sequences in an argument list. If escape sequences or other nonprintable characters are included in an argument list, they are passed as input to PATHMON and unexpected results might occur. Hewlett Packard Enterprise recommends that all characters specified for the ARGLIST attribute be ASCII printable. To specify escape sequences in an argument list, Hewlett Packard Enterprise recommends that you use the SPI management programming interface, see the *TS/MP 2.5 Management Programming Manual*.

If an argument includes blanks, quotes, commas, or semicolons, use the same rules as those for presenting names and values in PATHCOM, and enclose the entire argument in quotes. For example, to specify these arguments:

```
arg 1
"Argument2"
argument3
argument "4"
```

you enter the command as follows:

```
SET SERVER ARGLIST "arg 1", """Argument2""",     &
                   argument3, "argument ""4"""
```

NOTE: Due to the extra characters required by PATHCOM to embed blanks, quotes, commas, and semicolons, it is possible for the input buffer to be larger than the 24,000-character maximum. In such cases, PATHCOM allows 500 additional bytes in the input buffer.

A null argument is specified using either a comma (if the argument is not the last argument in the list), or double quotes. See the examples later in the discussion of this command.

For a discussion of when to use parentheses, see the subsection **Command Format** on page 134.

This attribute is valid for OSS server processes only.

Table Continued

ASSIGN **logical-unit , assign-spec**

assigns a logical file name to a physical Guardian file and specifies attributes of the file. The PATHMON process stores the values assigned by this attribute and sends the values to the server process using an ASSIGN message when the server process is opened.

NOTE: ASSIGN values are stored in a configuration file in alphabetic order; they are also accessed in alphabetic order.

The parentheses can be omitted when this attribute is the last one in the SET SERVER command.

This attribute is valid for Guardian server processes only.

If the server program is written in COBOL, this attribute is used to override the program file assignment made in the SELECT clause.

logical-unit

specifies the name assigned to a Guardian file and file attributes.

NOTE: Logical unit of ASSIGN must start with an alphabet.

program-unit

specifies the name of the server in the source program. In COBOL, this is the PROGRAM ID name. The program name can be different from the file name for the source code or for the run unit. If the server is a subprogram, this name is used in the CALL statement.

logical-file-name

specifies the name of the file in the server program. **logical-file-name** is associated with a physical file. For example, in a COBOL program **logical-file-name** is the file description name in a SELECT clause.

* replaces **program-unit** for the server with the file name specified in the PROGRAM attribute.

Table Continued

assign-spec

specifies values for the physical file.

file-name

specifies the name of the Guardian file, which represents the physical file assigned to the server program; this name can be a DEFINE name.

If you omit this option, no value is passed for this field of the ASSIGN message.

Defaults for file name expansion are based on values you specify for the CMDVOL command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

create-spec

specifies one of these file creation attributes or open attributes:

File-creation or Open Attribute	Description
EXT	Specifies the size of the file extents allocated to the file. **pri-extent-size** and **sec-extent-size** can be integers from 1 through 65,535.
EXCLUSIVE SHARED PROTECTED	Specifies the exclusion mode for **logical-unit**. The exclusion mode determines how other processes can access the file identified in **logical-unit**.
I/O INPUT OUTPUT	Specifies the access mode for **logical-unit**. The access mode designates what file operations can be performed.
CODE file-code	Assigns a file code to **logical-unit**.**file-code** can be an integer 0 through 65535. If you omit file-code, the value is 0.
REC record-size	Specifies the length of the records in logical-unit. record-size can be an integer from 1 through 4072.
BLOCK block-size	Sets the size of the data blocks for **logical-unit**. block-size can be an integer from 1 through 4096.

For more information about the ASSIGN attribute, see the ASSIGN command in the *TACL Reference Manual*.

Table Continued

AUTORESTART **number**

specifies the number of times that the PATHMON process attempts to restart a server process within a fixed 10-minute interval after an abnormal termination, such as a call to the Guardian ABEND procedure.

number can be a value from 0 through 32,767. If you omit this attribute, the default is 0.

After an abnormal failure of a server process, the action caused by this option is equivalent to using the START SERVER command and is effective for static server processes only.

This attribute is valid for Guardian and OSS server processes.

Table Continued

CPUS **primary:backup** | **cpu** | **cpu** (**cpu-wt**)

specifies the list of processors in which the server processes in this server class run. The processors can be specified either as primary and backup processor pairs, or as a list of single CPUs; the two types of lists cannot be mixed.

primary:backup

The PATHMON process uses this sequence of pairs in the order specified as it creates the server processes. You can define a maximum of 16 pairs of processor numbers; processor numbers for backup CPUs need not be unique.

If a primary processor is down, the PATHMON process exchanges the roles of the primary and backup CPUs. For example, if you define these:

```
CPUS (0:1, 2:3, 4:5)
```

and processor 2 is down, the PATHMON process starts the servers in CPUs 0, 3, and 4.

cpu

The PATHMON process uses this sequence of individual CPUs in the order specified as it creates the server processes. A processor can be specified only once in the list; you can define a maximum of 16 processor numbers. The BACKUPCPU attribute is not included in a PARAM message if a single processor list includes only one processor.

If a processor is down, the PATHMON process uses the next processor in the list. For example, if you define these:

```
CPUS (1, 3, 5)
```

and processor 1 is down, the PATHMON process starts the next server in processor 3.

cpu (cpu-wt)

is the weight defined for the CPU. PATHMON stores the CPU weights in the PATHCTL file. It is used to create a new process.

The processes are created on a CPU such that the percentage of processes running on a CPU to the total number of processes on all CPUs is almost equal to the defined weights. The CPU weight must be in the range of 1 through 100. The sum of all CPU weights must be 100.

NOTE: Backup CPU cannot be specified when using CPU weights.

If you omit this attribute, the PATHMON process uses all available CPUs on the system as primary and backup processor pairs according to an internally defined algorithm, and sends the BACKUPCPU param.

Only Guardian server processes can be run as NonStop processes. Therefore, for Guardian server processes only, the PATHMON process passes to the server process a PARAM message containing BACKUPCPU filled in with a 3-byte ASCII representation of the processor number of the backup processor. Use the GETBACKUPCPU procedure to retrieve the backup processor number from the PARAM message. (For more information on the GETBACKUPCPU procedure, see the *COBOL Reference Manual*.) The server process can then use the backup processor as an alternate processor.

When this attribute is included as part of the PROCESS attribute specification, and at server class level it is *CPU* or *primary:backup* , the PATHMON process uses the CPUs specified for the server process. The CPUs specified for the server process do not need to match the CPUs specified for the

Table Continued

server class. In case of *cpu-wt* , CPUs specified for server class override the effect of CPU configured at server process level.

This attribute is valid for Guardian and OSS server processes.

```
CREATEDELAY number { HRS | MINS | SECS | CSECS }
```

specifies the maximum amount of time a link manager (that is, a TCP, or ACS subsystem processes) waits to use an established link to a server class before requesting a new link from the PATHMON process that controls the server class.

CREATEDELAY must be a value within one of these ranges: 0 through 16,383 CSECS 0 through 16,383 SECS 0 through 1092 MINS 0 through 18 HRS.

If you omit this attribute, the default value is one MIN.

If the link manager has no established links or it calculates that a static link is available, the link manager immediately requests a link without waiting for CREATEDELAY to expire. If an established or newly created link becomes available while the link manager is waiting for CREATEDELAY to expire, the link manager immediately uses that link.

When the TCP calculates that no static link is available, it delays all future link requests to the PATHMON process by the value of CREATEDELAY. If CREATEDELAY is not specified, a default delay of 60 SECS is imposed.

NOTE: To minimize internal TCP or ACS subsystem queuing of requests, or if subsecond response time is required, set CREATEDELAY to zero or a very small value in seconds. This configuration increases the likelihood of additional server process creation during transaction peaks.

If ACS subsystem processes are unable to implement a CREATEDELAY timer due to insufficient system resources, it assumes CREATEDELAY is 0 and immediately requests a link from the PATHMON process.

This attribute is valid for Guardian and OSS server processes.

```
CWD oss-pathname
```

specifies the absolute OSS pathname of the current working directory of an OSS server process. This value is used to resolve relative pathnames specified for other OSS server process attributes in the server class. Case is significant. For a discussion, see **OSS Pathnames** on page 137.

If you omit this attribute in the SET command, relative OSS pathnames are resolved using the default directory specified by the CMDCWD command. If no value is set for this attribute via either command, there is no default.

This attribute is valid for OSS server processes only.

Use the OSS getcwd() function to obtain the name of the current working directory.

Table Continued

DEBUG { ON | OFF }

specifies whether the server processes in this server class enter debug mode when starting. If you omit this attribute, the default is OFF.

ON

directs the server processes to enter debug mode. If the PATHMON process was started with INSPECT ON or the server program was compiled or bound with the Inspect process defined, the servers enter Inspect mode for debugging. For more information about Inspect mode, see the *Inspect Manual*. For information regarding errors that can occur when a server process is in debug mode, see the *TS/MP 2.5 Pathsend and Server Programming Manual*.

OFF

directs the server processes not to enter debug mode.

This attribute is valid for Guardian and OSS server processes.

DEFINE **define-name, define-attribute-spec**

assigns a DEFINE definition as part of a server class definition. When a server process in the server class is created, the specified DEFINEs are included in the process context.

define-name

specifies a valid name to be assigned to the DEFINE. A DEFINE name can be from 2 through 24 characters in length. The first character must be an equal sign (=) and the second must be a letter. The name can contain alphanumeric characters, hyphens (-), underscores (_), and circumflexes (^).

define-attribute-spec

specifies the name of a valid DEFINE attribute and the value you associate with it.

This attribute is valid for Guardian and OSS server processes.

For detailed information about DEFINEs, see the *TACL Reference Manual and the Guardian User's Guide*.

DELETEDELAY **number** { HRS | MINS | SECS }

specifies the maximum amount of time a link between a link manager that is ACS subsystem processes or TCP, and a dynamic server process in a server class can remain idle before the link manager automatically returns the link to the dynamic server.

number must be a value within one of these ranges:

0 through 16,383 SECS 0 through 1092 MINS 0 through 18 HRS.

If you omit this attribute, the default is 10 MINS.

If ACS subsystem processes are unable to implement a DELETEDELAY timer due to a system resource problem, the ACS subsystem processes assume DELETEDELAY is 0 and returns the link to the PATHMON process.

This attribute is valid for Guardian and OSS server processes.

Table Continued

`ENV` **`name=value`**

specifies one or more OSS environment variables sent to OSS server processes in the server class. Multiple environment variables can be specified in a single command as a list of name-value pairs, separated by commas, or each variable can be specified in a separate `SET SERVER ENV` command. Within a server process, environment variables are available as an array of null-terminated strings, ending with a null pointer, pointed to by the external variable environ [].

You can specify from 3 through 24,000 characters for the ENV attribute. The length is calculated as the number of bytes of storage needed (the length after any quotes is removed and a null byte added for each environment variable). However, the total length of the values set for all ARGLIST and ENV values must not exceed 24,000 characters. The string value can be a null value. If you omit this attribute, no environmental information is sent to the server processes in this server class.

△ **CAUTION:** Neither PATHCOM nor the PATHMON process checks for escape sequences in an environment list. If escape sequences or other nonprintable characters are included in an environment list, they are passed as input to PATHCOM, and unexpected results might occur. Hewlett Packard Enterprise recommends that all characters specified for the ENV attribute be ASCII printable. To specify escape sequences in an environment list, Hewlett Packard Enterprise recommends that you use the SPI management programming interface; see the *TS/MP 2.5 Management Programming Manual*.

If a name or value includes blanks, quotes, commas, or semicolons, use the same rules as those for presenting names and values in PATHCOM (the entire list must be enclosed in quotes.) However, PATHCOM does not use blanks directly before or after the equal sign (=) if they are not enclosed in quotes. For examples, see the description of the ARGLIST attribute.

In addition, environment variable names (**`name`**) cannot contain an equal sign (=). (The first equal sign PATHCOM encounters is used as the delimiter for the environment variable name.) However, an equal sign can be included in the value. For example, to specify an environment variable named env1 having a value of abc=cba, enter the command as follows:

```
SET SERVER ENV env1=abc=cba
```

For a discussion of when to use parentheses, see the subsection **Command Format** on page 134.

This attribute is valid for OSS server processes only.

NOTE: Due to the extra characters required by PATHCOM to embed blanks, quotes, commas, and semicolons, it is possible for the input buffer to be larger than the 24,000-character maximum. In such cases, PATHCOM allows 500 additional bytes in the input buffer.

Table Continued

GUARDIAN-LIB **file-name**

specifies the name of the user library object file that contains additional server code space and procedures that can be shared among servers. 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 server code. For example:

```
SET SERVER PROGRAM $DATA.WORK.SRVOBJ
SET SERVER GUARDIAN-LIB $DATA.WORK.SRVLIB
```

In this example, whoever starts the PATHMON process must have WRITE and EXECUTE access to the server object file.

If you omit this attribute, the operating system accesses all of the server code, including the user library object file, through the file that you define with the PROGRAM attribute.

Defaults for file name expansion are based on values you specify for the CMDVOL command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

After starting a server process, the operating system modifies the server object file to point to the user library object file.

You can set the server user library file using the TACL RUN command or using the Binder program development tool. For more information, see the *Binder Manual*.

This attribute is valid for Guardian and OSS server processes.

HIGHPIN { ON | OFF }

specifies whether the server process runs at a high PIN or a low PIN:

ON	The server process runs at a high PIN.
OFF	The server process runs at a low PIN.

If you omit this attribute, the default is OFF.

To run a server at a high PIN, you must also set the ?HIGHPIN compiler directive in your server source code file. For more information about the ?HIGHPIN compiler directive, see the appropriate programming language reference manual.

This attribute is valid for Guardian and OSS server processes.

HOMETERM **file-name**

specifies the name of the home terminal for servers in the server class.

If you omit this attribute, the default is the home terminal of the PATHMON process.

Defaults for file name expansion are based on values you specify for the CMDVOL command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

This attribute is valid for Guardian and OSS server processes.

Table Continued

IN **file-name**

specifies the name of the input file passed to the server process in the startup message; this can be a DEFINE name.

System defaults for file name expansion are based on values you specify for the CMDVOL command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

If you omit this attribute, spaces are passed to the servers for the file name.

This attribute is valid for Guardian server processes only.

LIKE **server-class**

sets the attribute values for the server class to those of the named server class. **server-class** must be the name of a previously added server class.

This attribute is valid for Guardian and OSS server processes.

LINKALGORITHM **link-algorithm-value**

specifies the value of the algorithm which PATHMON process uses to select link grant algorithm for granting link. Valid values for this attribute are REQUESTERCPU and DEFAULT. For REQUESTERCPU, link grant algorithm gives preference to static server processes running on the CPU or configured with the CPU same as that of the requester.

LINKDEPTH **number**

specifies the maximum number of concurrent links that any one link manager that is the ACS subsystem processes, or a TCP, can have to a given server process before the PATHMON process directs that link manager's link requests to another server within the server class.

The value of LINKDEPTH must not exceed the value of MAXLINKS and cannot exceed 255. In the unlikely event that if MAXLINKS is set to 0, set the value of LINKDEPTH to 1. For information about configuration limits, see **Configuration Limits and Defaults** on page 364.

If you omit this attribute, the default is 1.

Typically, you use the LINKDEPTH attribute to multithreaded servers; for servers that are not multithreaded, use the default of 1 for LINKDEPTH.

This attribute is valid for Guardian and OSS server processes.

Table Continued

MAXLINKS **number**

specifies the maximum number of concurrent links permitted between all ACS subsystem processes and TCPs and a server in a server class.

number must be a value from 0 through 4095. This attribute establishes the maximum number of concurrent send operations to a single server process. The TCP or ACS subsystem processes do not check how many links a server has with other TCPs or ACS subsystem processes.

If the value for this attribute is too large, it can cause the requests to the server process to be queued at the server for an unacceptable length of time. For example, if MAXLINKS is equal to 20, there can be 20 concurrent requests outstanding to a server. If the transaction service time is 1 second, a response time of more than 20 seconds can occur.

If you omit this attribute, the default is an unlimited number of links. Hewlett Packard Enterprise strongly recommends that you set a value for MAXLINKS.

NOTE: The value of MAXLINKS and the value of the TABLE OCCURS clause in the RECEIVE–CONTROL paragraph of your COBOL server programs must correspond with each other. The MAXLINKS value must be equal to or less than the TABLE OCCURS value. When the COBOL RECEIVE–CONTROL table is full, new requests to the server fail with file-system error 12 (File in use). This error is most likely to occur during peak transaction load times. For more information about the RECEIVE–CONTROL paragraph of a COBOL program, see the *COBOL Reference Manual*.

If this server is an associative server, then you must include links allocated by other PATHMON environments when calculating the size of the RECEIVE-CONTROL paragraph.

The value of MAXLINKS must always be greater than or equal to the value of LINKDEPTH (unless MAXLINKS is set to 0).

This attribute is valid for Guardian and OSS server processes.

MAXSERVERS **number**

specifies the maximum number of server processes in this server class that can run at the same time. **number** must be a value from 0 through 4095. If you omit this attribute, the default is 1.

If MAXSERVERS is greater than the number of server process names that you define with the PROCESS attribute, the operating system assigns process names in the form $X*nnn*, $Y*nnn*, or $Z*nnn* after the last predefined process name is used.

The sum of all MAXSERVERS values for all server classes cannot exceed the number defined in the SET PATHWAY MAXSERVERPROCESSES attribute.

This attribute is valid for Guardian and OSS server processes.

Table Continued

NUMSTATIC **number**

specifies the maximum number of static servers within this server class. These servers can be started using the START SERVER command or as a result of a link request. These servers remain running until a STOP SERVER command is issued or all of their links are returned by link managers (that is, TCPs or ACS subsystem processes).

number must be a value from 0 through 4095. If you omit this attribute, the default is 0 (that is, all servers are dynamic). The PATHMON process does not create a server until it receives a link request from a TCP or ACS subsystem processes. When NUMSTATIC is 0, the first link to a server class counts as a static link in the STATS command display.

The number for MAXSERVERS minus the number for NUMSTATIC is the number of dynamic servers for the server class. Links to dynamic servers are granted to a TCP or ACS subsystem processes by the PATHMON process when a link request cannot be satisfied by a static server. Dynamic server processes are started by the PATHMON process only as the result of a link request; they are not started by the START SERVER command.

The value of NUMSTATIC cannot exceed the value of MAXSERVERS.

This attribute is valid for Guardian and OSS server processes.

OUT **file-name**

specifies the name of the OUT file in the startup message; this can be a DEFINE name.

Defaults for file name expansion are based on values you specify for the CMDVOL command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

If you omit this attribute, spaces are passed to the servers for the file name.

This attribute is valid for Guardian server processes only.

Table Continued

OWNER **ownerid**

specifies the user ID that controls access from a Pathsend process to a server class. (The TCPs ignore this server attribute.) The user ID must be known to the system in which PATHCOM is running. Use this attribute in conjunction with the SECURITY attribute.

The value of **ownerid** is one of these:

```
[ \system-number .| \*.] group-number, user-number
[ \node . | \* .] group-name . user-name
```

If the NODEINDEPENDENT attribute of the 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 value is whatever node the PATHMON process is running on after cool start. For more on the * name, see the description of **SET PATHWAY Command**.

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 PATHMON process's node.

If you omit the **group-name** and **user-name** values, they default to the user ID values of the user who started the PATHMON process. (This attribute does not use the defaults established by the CMDVOL command).

This attribute is valid for Guardian and OSS server processes.

PARAM **parameter-name parameter-value**

assigns a string value to a parameter name. You can omit the parentheses if this parameter is the last one of the SET SERVER command.

parameter-name

specifies the name to which you assign **parameter-value** . The name must begin with an alphabetic character.

parameter-value

specifies a character string value that you assign to **parameter-name** . This string must be enclosed in quotation marks if you include blanks, commas, or quotation marks in the value. Embedded quotation marks must be doubled.

The PATHMON process uses the information you supply for this parameter to send a PARAM message to each server that it starts. No more than 120 PARAM messages can be added to any one server class. In practice, the actual number might be less because all individual PARAMs for a server are collected into a single PARAM message, limited in length to 1028 bytes.

For more information about PARAM messages, see the **Guardian User's Guide**. For more information on this parameter, see the *COBOL Reference Manual and TACL Reference Manual*.

This attribute is valid for Guardian server processes only.

Table Continued

PRI **priority**

specifies the priority at which the servers of this server class run.

priority can be a value from 1 through 199. If you omit this attribute, the default is 10 less than the priority of the PATHMON process.

This attribute is valid for Guardian and OSS server processes.

Table Continued

```
PROCESS $process-name [ process-attribute ]
```

assigns server names and overrides corresponding attributes of the `SET SERVER` command. The PATHMON process gives the process names to the servers when it creates them. It assigns the names in the order that you define them in the server class description.

This attribute is valid for Guardian and OSS server processes.

`$`**process-name**

specifies the name of a server process within the server class.

process-name can be a maximum of six characters, beginning with a dollar sign ($) and followed by one to five alphanumeric characters. The first character must be a letter. A secondary qualifier for the process name is allowed. The secondary qualifier can be a maximum of 8 characters beginning with a # (pound sign) and followed by one to seven alphanumeric characters. The first character must be a letter.

If you omit this attribute, the default is a process name that the operating system assigns in the form $X**nnn** , $Y**nnn** , or $Z**nnn** .

For better manageability, assign explicit server process names instead of letting the operating system assign the server process names.

process-attribute

specifies any of these process attributes:

```
ASSOCIATIVE { ON | OFF }
```

specifies whether the PATHMON process in this Pathway environment grants links to a server process that already exists in another Pathway environment.

If you omit this attribute, the default is OFF.

ON	s the PATHMON process to grant links to the server process, even if a server by the same name already exists. A `START SERVER` command is successful.
OFF	does not the PATHMON process to grant links to the server process if a server by the same name already exists. A `START SERVER` command fails.

```
CPUS primary:backup | cpu
```

specifies the primary and backup processors in which the server process runs or a list of individual CPUs.

When this attribute is included as part of the PROCESS attribute specification, the PATHMON process uses the CPUs specified for the server process. The CPUs specified for the server process do not need to match the CPUs specified for the server class.

Table Continued

If you omit this attribute, the PATHMON process assigns all CPUs. See the description of the CPUS attribute earlier in this command.

`DEBUG { ON | OFF }`

determines whether the server process enters debug mode upon startup.

If you omit this attribute, the default is ON.

ON	directs the server process to enter debug mode.
OFF	directs the server process not to enter debug mode.

`GUARDIAN-SWAP $`**`volume`**

specifies the Guardian name of the disk volume for the swap file formerly used by the operating system for memory swaps of the server data area. Allocation of this swap file is now handled by the Kernel Managed Swap Facility (KMSF) on behalf of the PATHMON process, and any value specified for this parameter is not used. However, if you specify a value for this parameter, it must represent an existing device, or an error occurs.

If you allow this parameter to default, recall that defaults for Guardian file name expansion are based on values you specify for the `CMDVOL` command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

`HOMETERM` **`file-name`**

specifies the name of the home terminal for the server process. See the description of the HOMETERM attribute earlier in this command.

`PRI number`

specifies the priority at which the server process runs. See the description of the PRI attribute earlier in this command.

`PROGRAM` **`file-name`** | **`oss-pathname`**

specifies the name of the object file to be used by the servers in this server class.

For Guardian processes, specify a Guardian file name. PATHCOM validates the file name when the `SET SERVER` command is issued.

Defaults for Guardian file name expansion are based on values you specify for the `CMDVOL` command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in `CMDVOL` Command and **SET PATHWAY Command** on page 171.

For OSS processes, specify an OSS pathname. If the CWD attribute is not specified (in the `CMDCWD` command or the `SET SERVER CWD` command), the OSS object file name must be an absolute OSS pathname. Case is significant. For a discussion, see **OSS Pathnames** on page 137. The PATHMON process validates the pathname when the `ADD SERVER` command is issued.

This attribute is required and is valid for Guardian and OSS server processes.

Table Continued

SECURITY **security-attribute**

specifies the users, in relation to the OWNER attribute, who can access a server class from a Pathsend requestor. (TCPs ignore this attribute).

The security attributes are the same as the Guardian file 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”. This attribute is valid for Guardian and OSS server processes.

STARTUP **string**

specifies the character string that is sent to a Guardian server process in the startup message. Leading blanks are ignored. If you want embedded blanks to be read as part of the startup string, enclose **string** in quotation marks. If the string includes quotation marks, they must be doubled.

This attribute is valid for Guardian server processes only.

STDERR **oss-pathname**

specifies the name of the OSS stderr (standard error) file to be used by the OSS servers in this server class. This file is opened only for write operations.

If the CWD attribute is not specified (in the CMDCWD command or the SET SERVER CWD command), the OSS stderr file must be an absolute OSS pathname. Case is significant. For a discussion, see **OSS Pathnames** on page 137. If the file specified does not exist, it is created. Output is appended to the end of the file. If you omit this attribute, a null string is sent to the server process.

This attribute is valid for OSS server processes only.

Table Continued

STDIN **oss-pathname**

specifies the name of the OSS stdin (standard input) file to be used by the servers in this server class. This file is opened only for read operations.

If the CWD attribute is not specified (in the CMDCWD command or the SET SERVER CWD command), the OSS stdin file must be an absolute OSS pathname. Case is significant. For a discussion, see **OSS Pathnames** on page 137.

If the file specified does not exist, it is created. If you omit this attribute, a null string is sent to the server process.

This attribute is valid for OSS server processes only.

STDOUT **oss-pathname**

specifies the name of the OSS stdout (standard output) file to be used by the servers in this server class. This file is opened only for write operations.

If the CWD attribute is not specified (in the CMDCWD command or the SET SERVER CWD command), the OSS stdout file must be an absolute OSS pathname. Case is significant. For a discussion, see **OSS Pathnames** on page 137. If the file specified does not exist, it is created. Output is appended to the end of the file. If you omit this attribute, a null string is sent to the server process.

This attribute is valid for OSS server processes only.

TIMEOUT **number** { HRS | MINS | SECS }

specifies the timeout value that a link manager such as a ACS subsystem process or TCP uses for the I/O operation to the server process.

TIMEOUT must be a value from 0 through 18 HRS, 0 through 1092 MINS, or 0 through 16,383 SECS. If you omit this attribute, there is no timeout on the server I/O operation. That is, the link manager waits indefinitely for the server I/O operation to complete.

A SERVERCLASS_SEND_ call that times out returns Pathsend error FESCSERVERLINKCONNECT (904) and file-system error 40. A SCREEN COBOL SEND operation that times out has a server I/O error 12 reported in the SCREEN COBOL TERMINATION-STATUS special register.

The SET SERVER TIMEOUT attribute is not active until the first open procedure has completed. The initial open of a server process by a ACS subsystem processes have an automatic 5 minute timeout period; the initial open of a server process by a TCP has an automatic 10 minute time-out period.

This attribute is valid for Guardian and OSS server processes.

For more information on Error 904.40, see **Usage Considerations for SET SERVER TIMEOUT Attribute** on page 244.

Table Continued

```
TMF { ON | OFF }
```

specifies whether servers in this server class can lock and update data files audited by the TMF subsystem.

If you omit this attribute, the default is OFF.

ON	s servers to lock and update audited files. The ACS subsystem processes or TCP process opens the servers with a syncdepth of 0.
OFF	does not allow servers to lock and update audited files. The ACS subsystem processes or TCP process opens the servers with a syncdepth of 1.

The ACS subsystem processes do not perform checkpoint operations; it automatically retries I/O errors in the same way as the operating system retries them. The TCP performs checkpoint operations, but does not automatically retry I/O errors; this is done by the operating system.

This attribute is valid for Guardian and OSS server processes.

Table Continued

UMASK Octalnumber

PATHMON process uses this attribute as a file mode creation mask while starting the server process. This attribute is applicable for OSS server classes. The Octalnumber entry is a three digit octal number that specifies the default permissions for the owner, group, and others.

When this command is executed PATHMON process defines a value for the UMASK attribute for the server process to be started. Each character in the UMASK value must be a valid octal number.

The UMASK permission combinations are as follows:

Octal number	Maximum allowed permissions	Description
0	r w x	Read, write, and execute.
1	r w -	Read and write.
2	r - x	Read and execute.
3	r - -	Read.
4	- w x	Write and execute.
5	- w -	Write.
6	- - x	Execute.
7	- - -	No permissions granted.

For example, if you specify a user mask (UMASK) of 037:

1. Owner has all permissions requested by the program creating the file (0).
2. Group has read permissions, but write and execute permissions are not allowed (3).
3. Others do not have any permissions (7).

For more information about UMASK, see *Open System Services System Calls Reference Manual*.

NOTE: The UMASK command can restrict permissions. The UMASK command cannot grant extra permissions beyond what is specified by the program that creates the file.

VOLUME **volume-spec**

For GUARDIAN processes, specifies the server class default volume and subvolume names to be passed to the server process in the startup message. This value is also used by the PATHMON process for the VOLUME attribute of _DEFAULTS DEFINE.

For OSS processes, specifies the server class default Guardian volume and subvolume used by the PATHMON process for the VOLUME attribute of the _DEFAULTS DEFINE.

Defaults for file name expansion are based on values you specify for the CMDVOL command and the SET PATHWAY NODEINDEPENDENT attribute. See the command descriptions in **CMDVOL Command** on page 145 and **SET PATHWAY Command** on page 171.

This attribute is valid for both Guardian and OSS server processes.

For more information on _DEFAULTS DEFINE, see **Server Class DEFINEs** on page 228.

Considerations

- When the relevant procedure calls (umask(2)) or shell utility commands (umask(1)) sets a UMASK, then this UMASK is valid. The files created by the process with this UMASK have the appropriate permissions irrespective of the Pathway settings.
- When the relevant procedure calls (umask(2)) or shell utility commands (umask(1)) does not set a UMASK and in the Pathway environment the UMASK attribute is set for a serverclass, then this UMASK is valid for the running server processes. The files created by these server processes have the permissions according to the UMASK attribute for the serverclass irrespective of the DUMASK existing in this Pathway environment.
- When the relevant procedure calls (umask(2)) or shell utility commands (umask(1)) does not set a UMASK and in the Pathway environment the UMASK attribute is not set for a serverclass but a DUMASK (default umask) is set, then this default UMASK is valid. The files created by the processes of this serverclass without a set UMASK have the permissions according to the DUMASK.
- When the relevant procedure calls (umask(2)) or shell utility commands (umask(1)) does not set a UMASK and in the Pathway environment the UMASK attribute is not set for a serverclass or a DUMASK (default umask) is not set, then the initial UMASK (`000`) of the operating system is valid. The value `000` means `rwx rwx rwx` or no restrictions. The files created by the processes that did not have a UMASK or get a UMASK by the Pathway have all the permissions.
- The operating system initializes a UMASK with the value `000` (default). This value remains valid until the process or a Pathway serverclass attribute sets a UMASK.
- The default value of the DUMASK in a Pathway is a nonexistent DUMASK. In this case, the process or a Pathway serverclass attribute that sets the UMASK is valid. If not set, then the default value (`000`) is valid.
- The default value of UMASK in the Pathway is nonexistent. In this case, the process that sets the UMASK is valid. If it is not set, then a set DUMASK or the default value (`000`) is valid.

Server Class DEFINEs

A server process context includes any DEFINEs that you specified using the DEFINE attribute of the `SET SERVER` command and a _DEFAULTS DEFINE automatically generated by the PATHMON process for values you did not specify. You can create DEFINEs to specify the default volume and subvolume for a server process and to specify files used by the server.

For example, these PATHCOM commands form part of a Guardian server class definition:

```
SET SERVER PROGRAM $DATA.PW.SRVOBJ
SET SERVER DEFINE =_DEFAULTS, CLASS DEFAULTS, &
                              VOLUME $VOL.SVOL &
                              CATALOG $DATA.MYCAT
SET SERVER DEFINE =EMPLOYEE, CLASS MAP, FILE \SYS.$D.APPL.EMP
ADD SERVER EMP-SERVER
```

You can specify the default volume and subvolume for a server class using the VOLUME attribute of the `SET SERVER` command or the VOLUME attribute of _DEFAULTS DEFINE. If you do not specify _DEFAULTS DEFINE, the PATHMON process uses the volume and subvolume you assigned with the VOLUME attribute. If you assign _DEFAULTS DEFINE and a VOLUME attribute, the PATHMON process ensures that these values correspond. If you add or replace _DEFAULTS DEFINE, the VOLUME value

changes. Similarly, altering the value of the VOLUME attribute changes the VOLUME attribute of _DEFAULTS DEFINE.

For more information about DEFINEs and using _DEFAULTS DEFINE, see the *TACL Reference Manual and the Guardian User's Guide*.

Considerations

- If you repeat a `SET SERVER` command with a different attribute value, the PATHMON process uses the last value entered for the server class attribute.
- All attributes defined for a server class must be consistent with the value defined for PROCESSTYPE. Inconsistent attributes are rejected with error 2066 or error 2067. To reduce possible confusion, Hewlett Packard Enterprise recommends that you configure the PROCESSTYPE attribute first.
- The `CMDCWD` and the `SET SERVER CWD` commands both define a value for the CWD attribute. They do not function the same way, however.

  When resolving relative OSS pathnames, the value defined by the `SET SERVER CWD` command takes precedence. The `RESET SERVER` command resets this value to null.

  The value defined by the `CMDCWD` command is used to resolve relative OSS pathnames only if no value is set using the `SET SERVER CWD` command. It is a default absolute pathname that you can define for a server class. The `RESET SERVER` command has no effect on this value. To reset this value to null, use the `RESET CMDCWD` command.

- A running server process that is stopped and restarted does not reuse the processor of the stopped server process. The restarted process uses the next processor specified in the cyclical list of CPUs.

  If you alter a server process, that process uses the first processor in the cyclical list of CPUs when it is restarted. If the server class is stopped and restarted, it does not reuse the processor of the stopped server class. The restarted server class uses the next processor in the cyclical list of CPUs.

- If you specify a value for the TIMEOUT attribute and an I/O to a server process times out because of the TIMEOUT value, the link management process cancels the I/O to the server and returns the link. If the link was the last link to the server, the server stops.
- For information about configuring the NonStop Translation Server for NonStop TUXEDO, see the Pathway Translation Server for the *NonStop TUXEDO System Manual*.

Examples

These commands define a subset of the server class attributes:

```
RESET SERVER

SET SERVER PROCESSTYPE GUARDIAN
SET SERVER ASSIGN DDL, \*.$DATA.DIC.DDL
SET SERVER AUTORESTART 4
SET SERVER CPUS (0:1, 2:3)
SET SERVER CREATEDELAY 10 SECS
SET SERVER DELETEDELAY 5 MINS
SET SERVER LINKDEPTH 1
SET SERVER MAXLINKS 5
SET SERVER MAXSERVERS 7
SET SERVER NUMSTATIC 2
SET SERVER OWNER \*.8,8
SET SERVER PARAM SWITCH-1 ON
SET SERVER PROGRAM INVENT
```

```
SET SERVER SECURITY "N"
SET SERVER STARTUP "MAX^POCB 80,RCV^LCBS 2"
SET SERVER TIMEOUT 15 MINS
SET SERVER TMF ON
SET SERVER VOLUME $MFG.SERV
SET SERVER IN \*.$RM123
SET SERVER OUT \*.$RM123

ADD SERVER SERV-ABC
```

Note that in the above example, the generic node name * is specified for the node names for the ASSIGN, OWNER, IN, and OUT file names. The * name designates the node where the PATHMON process is currently running. In this example, the OWNER, IN, and OUT file names are stored in the PATHMON configuration file using * as the node name, making these attributes node-independent. For more information on configuring for node independence, see **SET PATHWAY Command** on page 171. Special care must be taken when migrating a Pathway environment to a new node to ascertain that certain attributes, such as HOMETERM, can migrate without problems. For guidelines, see **Migrating Your Environment to a Different System** on page 109.

In the configuration set by these commands, the first two server processes are created with the names $X and $Y; all others have system-generated names. $X is created in processor 4 or 5, and $Y is created in processor 6 or 7. All other processes are created in processor 0, 1, 2, or 3. Only $Y is started in debug mode.

```
SET SERVER MAXSERVERS 5
SET SERVER CPUS ( 0:1, 2:3 )
SET SERVER DEBUG OFF
SET SERVER PROCESS $X ( CPUS 4:5 )
SET SERVER PROCESS $Y ( CPUS 6:7, DEBUG ON)
```

These `SET SERVER` commands show the use of parentheses to avoid ambiguity:

```
SET SERVER ( PARAM SWITCH-1 ON , SWITCH-15 OFF ) , &
         MAXLINKS 10
SET SERVER (ARGLIST "arg 1", """Argument2"""), PRI 141, &
        (ENV env1="env1", env2="here is env2"), MAXLINKS 5
```

These command specifies a list of five arguments; the second and fourth argument are null strings:

```
SET SERVER ARGLIST arg1,"",arg3,,arg5
```

These command specifies a list of five arguments; all arguments are null strings except for the third argument:

```
SET SERVER ARGLIST ,,arg3,,""
```

SHOW SERVER Command

Use the `SHOW SERVER` command to display a subset of the attribute values for a server class, in alphabetic order. Only attributes consistent with the process type (Guardian or OSS) are displayed. If a value has not been configured for an attribute, this command displays the current default value.

```
SHOW [ / OUT list-file / ] SERVER
```

OUT list-file

directs output to the named list file; this can 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.

Display Formats

When PATHCOM first starts or after a RESET SERVER command is issued, the SHOW SERVER command displays only the required SET SERVER command attributes and the server class default values. A question mark appears for required attribute values that you must set before adding the server class description to the PATHMON configuration file.

SHOW SERVER Default and Guardian Display on page 231 shows the default display the display returned when no SERVER object attributes are set. Because the default PROCESSTYPE is GUARDIAN, this is also the display for Guardian server processes. A question mark (?) indicates a required attribute.

SHOW SERVER Default and Guardian Display

```
SERVER
  PROCESSTYPE GUARDIAN
  AUTORESTART 0
  CREATEDELAY 1 MINS
  DEBUG OFF
  DELETEDELAY 10 MINS
  HOMETERM $term-name
  HIGHPIN OFF
  LINKDEPTH 1
  MAXSERVERS 1
  NUMSTATIC 0
  OWNER owner
  PRI priority
  PROGRAM ?
  SECURITY "N"
  TMF OFF
  VOLUME \node.$volume.subvolume
```

When the PROCESSTYPE attribute is set to OSS but no other values are defined, the same information appears.

For any attributes that are set to default to the node where PATHMON is running, the generic node name * appears for the node name for that attribute. Attributes would be set to default to the current PATHMON node, for example, if you had SET NODEINDEPENDENT to ON when you used the SET PATHWAY command to configure your environment. Attributes would also default to the current PATHMON node is you set the node default to * when using the CMDVOL command in your last PATHCOM session.

Additional Attributes Displayed

These Guardian attributes are displayed only if they have values defined:

```
ASSIGN       OUT
DEFINE       PARAM
GUARDIAN-LIB PROCESS
IN           STARTUP
```

These OSS attributes are displayed only if they have values defined:

```
ARGLIST GUARDIAN-LIB
CWD     PROCESS
DEFINE  STDERR
ENV     STDIN
        STDOUT
```

The CWD attribute appears when a value is set either by the CMDCWD command or by the SET SERVER CWD command. All OSS pathnames, argument lists, and environment variables are displayed as quoted strings, regardless of whether there are embedded quotes (similar to the way PARAM values are displayed).

Example

This command directs the output from the SHOW SERVER command to the file SERFILE:

```
SHOW/OUT SERFILE/SERVER
```

START SERVER Command

Use the START SERVER command to initiate operation of the static servers in a server class. This command does not affect server classes in the FROZEN state.

```
START
    { [ SERVER ] server-class [ , PROCESS $process-name ] }
    { [ SERVER ] ( server-class [ , server-class ]... ) }
    { SERVER * }
```

server-class

specifies the name of a server class. Use either a single name or several names separated by commas and enclosed in parentheses.

PROCESS $process-name

specifies the name of a single server in the server class.

SERVER *

starts all server classes in the PATHMON configuration file and in the correct state.

Considerations

- The START SERVER command starts the number of servers defined by the SET SERVER NUMSTATIC attribute for the server class.
- For information on starting associative servers, see “Associative Servers” under **SET SERVER Command** on page 204 earlier in this section.

Guardian Server Startup Protocol

The PATHMON process selects a processor pair from the list specified with the CPUS attribute of the SET SERVER command, or selects the processor specified with the PROCESS attribute of the SET SERVER command.

The startup protocol for servers consists of these steps:

1. The PATHMON process creates the server by calling the NEWPROCESS or PROCESS_CREATE_ procedure.

   The server process context includes all DEFINEs specified in the server class configuration (using the SET SERVER DEFINE command).

2. The PATHMON process sends the server a startup message.

The startup message has a header that is constructed from what you specified for the IN, OUT, and VOLUME attributes in the SET SERVER command. If you specified a character string for the STARTUP attribute, that is also included.

3. If the server replies with a value of 70 (FECONTINUE) to the startup message, the PATHMON process sends ASSIGN and PARAM messages to the server.

The ASSIGN messages are constructed directly from the information you provided for the ASSIGN attribute to the SET SERVER command. The PARAM message contains one item for each attribute value you specified for the PARAM attribute of the SET SERVER command. The number of the processor not chosen as the primary processor is reported in the BACKUPCPU attribute, which the PATHMON process adds to the PARAM message. For a description of how this value is chosen, see the information about CPUS attribute of the SET SERVER command.

Examples

This command starts all of the static servers defined for all server classes:

```
START SERVER *
```

This command starts only the static server processes defined for the named server classes:

```
START SERVER (IMF-CLASS, ORD-SRV, MFG-SRV)
```

This command starts a single process named $SLIB in the server class IMF-CLASS:

```
START IMF-CLASS, PROCESS $SLIB
```

STATS SERVER Command

Use the STATS SERVER command to display resource usage and performance statistics, including response time information. These statistics provide information about server class operations. They are tracked by the ACS subsystem processes and TCPs that have links to a server class.

The ACS subsystem processes collect these statistics continuously; you cannot set an attribute that controls ACS subsystem processes track statistics. The STATS attribute of the SET TCP command controls whether a TCP tracks its statistics.

NOTE: The ACS subsystem processes do not collect performance statistics. Performance statistics are collected only by the TCPs, which are configured in your environment only if you are running the Pathway/iTS product. For a discussion of statistics collections, see **Tuning Your System by Using Statistics** on page 120.

```
STATS [ / OUT list-file / ]

   { [ SERVER ] server-class }
   { [ SERVER ] ( server-class [ , server-class ]... ) }
   { SERVER * }

   [ , server-attribute [ , server-attribute ]... ]

server-attribute is:

    COUNT number
    FREQTABLE
    INTERVAL number { HRS | MINS | SECS }
    RESET
```

OUT list-file

directs output to the named list file; this can 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.

server-class

specifies the name of a previously defined and added server class. Use either a single name or several names separated by commas and enclosed in parentheses.

SERVER *

displays statistics for all server classes in the PATHMON configuration file.

COUNT number

specifies the number of times that the `STATS` command repeats. You can stop the command from repeating by pressing the **Break** key.

FREQTABLE

generates a frequency distribution table containing statistics for each server class.

NOTE: Only TCPs can collect frequency distribution information. You can display this information for a server only if your environment includes the Pathway/iTS product.

The frequency distribution table is not generated—even if you specify the `FREQTABLE` option—under these conditions:

- There are 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.

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

- The `STATS SERVER` command displays statistics for link manager that is the ACS subsystem processes. The link managers store the statistics for a server class. If the server class stops for any

reason, its statistics are lost. Consequently, when a server in that class is restarted, the link manager has no statistics for the server class.

- For more information about the STATS SERVER command display, see **Tuning Your System by Using Statistics** on page 120.

Examples

This command displays statistical information gathered about the specified server class, eight times, at one-hour intervals:

```
STATS CLASS-1, COUNT 8, INTERVAL 1 HRS
```

This command collects statistical information about all of the server classes for four times, at 15-minute intervals, and directs the information to a specified file:

```
STATS /OUT STATFLE/ SERVER *, RESET, INTERVAL 15 MINS,
COUNT 4
```

STATUS SERVER Command

Use the STATUS SERVER command to display the current status of a server class. This command displays status information for server classes linked to link manager that is the ACS subsystem processes.

```
STATUS [ / OUT list-file / ]
   { [ SERVER ] name [ , PROCESS $process-name ] }
   { [ SERVER ] name [ , PROCESSES             ] }
   { [ SERVER ] name [ , DETAIL                ] }
   { [ SERVER ] name [ , FREEZE                ] }
   { [ SERVER ] ( name [ , name ]... ) [ , PROCESSES  ] }
   { [ SERVER ] ( name [ , name ]... ) [ , DETAIL     ] }
   { [ SERVER ] ( name [ , name ]... ) [ , FREEZE     ] }
   { SERVER * [ , PROCESSES ] [ , DETAIL ] [ , FREEZE ] }
   { FREEZE }
```

OUT list-file

directs output to the file that you specify; this can 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.

name

specifies the name of a previously defined and added server class.

PROCESS $process-name

specifies the name of the server process associated with the designated server class.

PROCESSES

displays the status of all server processes associated with the specified server class.

DETAIL

includes status information on the servers processes associated with the specified server class.

FREEZE

displays the names of the server classes in the FROZEN or FREEZEPENDING state. For each server class that is in the FREEZEPENDING state, the names of the terminals that have not yet accepted

the freeze of the server class are also listed. The PATHMON process does not report FREEZE information about external link manager that is the ACS subsystem processes.

If the FREEZE option is specified with SERVER *, and no server classes are in the FREEZE-PENDING state, PATHCOM returns a prompt.

SERVER *

displays the status of all server classes defined for the Pathway environment.

STATUS FREEZE

displays the names of all server classes that are in the FROZEN or FREEZE-PENDING state.

Display Format Without DETAIL

Following is the format of the display returned by the STATUS SERVER command without the DETAIL option. This display is returned for these commands: STATUS SERVER, STATUS SERVER PROCESS, and STATUS SERVER PROCESSES. The fields are described in the following section:

STATUS SERVER Display Format Without Detail

```
SERVER          #RUNNING       ERROR       INFO
server-name      running      pw-error     info [freeze-state]
    .               .            .           .
    .               .            .           .
    .               .            .           .
```

The fields in this display are as follows:

Display Field	Description	
SERVER	Name of the server class.	
#RUNNING	Number of server processes in the server class that are currently running.	
ERROR	Pathway error number. This number includes any error that might occur when the PATHMON process attempts to create a dynamic server.	
INFO	For additional information regarding the error number, see the error messages listed in **PATHMON Messages (Numbers 1000-1499)** on page 245 through **LINKMON Log Messages** on page 306. The freeze-state might also appear. Possible values are:	
	FREEZE-PENDING	The server class received a FREEZE command, but the state of one or more terminals delays the freeze from completing. The STATUS SERVER FREEZE command can be used to display the names of these terminals.
	FROZEN	The server class is frozen. No communication can take place with any server of the server class until the class receives a THAW command.

Display Format With DETAIL

Following is the format of the display returned by the STATUS SERVER command with the DETAIL option. The fields are described in the following section:

STATUS SERVER Display Format With DETAIL

```
   SERVER        #RUNNING    ERROR    INFO
server-name       running   pw-error   info [freeze-state]
     .               .         .         .
     .               .         .         .
     .               .         .         .
   PROCESS       STATE       ERROR     INFO   #LINKS    WEIGHT
process-name     state      pw-error    info    links    weight
      .            .           .         .       .        .
      .            .           .         .       .        .
      .            .           .         .       .        .
```

The additional fields returned in this display are as follows:

Display Field	Description	
PROCESS	Guardian name of the server process.	
STATE	Current state of the server process. Possible values are:	
	STOPPED	The server process is stopped and has no links to link manager that is the ACS subsystem processes.
	RUNNING	The server process is running.
	PENDING	The last link was returned but the server process has not yet stopped. This value might indicate (1) that the PATHMON process has received a stop message but the link manager has not been notified the server process is terminated or (2) that the link manager has notified the PATHMON process that it is no longer using the server process, and the PATHMON process is awaiting a stop message.
ERROR	PATHMON error number. This number includes any error that might occur when the PATHMON process attempts to create a dynamic server.	
INFO	For additional information about the error, see the error messages listed in **PATHMON Messages (Numbers 1000-1499)** on page 245 through **LINKMON Log Messages** on page 306.	
LINKS	Number of the links from link managers to the server process.	
WEIGHT	Indicates the use of the server process compared to the use of other server processes in the server class. For each link to a server class, the server process's weight is incremented by 3 for the first link from a given link manager that is the ACS subsystem process, by 2 for the second link from that link manager, and by 1 for all subsequent links from that link manager. Note that whenever possible, a link manager uses older links before newer ones.	

NOTE: #LINKS value shows 0 if rebalancing of server processes is in progress. After the rebalancing is completed, the exact #LINKS value appears.

Display Format, PROCESSES Attribute, Without DETAIL

Following is the format of the display returned by the `STATUS SERVER PROCESS` or `STATUS SERVER PROCESSES` commands without the `DETAIL` option. The fields not already described are described in the following section:

STATUS SERVER PROCESS[ES] Display Format

```
=status server *, processes

SERVER           #RUNNING   ERROR  INFO
SRV1                 5

  PROCESS          STATE      ERROR  INFO     #LINKS  WEIGHT
  $Z29W            RUNNING                        0       0

  PROCESS          STATE      ERROR  INFO     #LINKS  WEIGHT
  $Z29X            RUNNING                        0       0

  PROCESS          STATE      ERROR  INFO     #LINKS  WEIGHT
  $Z29Y            RUNNING                        0       0

  PROCESS          STATE      ERROR  INFO     #LINKS  WEIGHT
  $Z29Z            RUNNING                        0       0

  PROCESS          STATE      ERROR  INFO     #LINKS  WEIGHT
  $Z2A0            RUNNING                        0       0
```

The additional fields returned in this display are as follows:

Display Field	Description	
LINKER	Name of a link manager, such as a LINKMON process, that is currently linked to the server.	
	LINKMON process names have the format: L**node.$process-name**	
	L	Identifies the process as a LINKMON process.
	\node	Name of the NonStop systemNonStop system on which the LINKMON process is running.
	$process-name	LINKMON process name. LINKMON process names are in the form $ZL**nn**, where **nn** is the number of the processor where the LINKMON process resides.
LINK COUNT	LINK COUNT is displayed when LINKDEPTH is greater than 1. *This field shows the number of links from the link manager to the serverclass.	
	LINK RANK: LINK RANK is displayed when LINKDEPTH is equal to 1.	

* For TS/MP 2.1 and later versions, the link is selected depending on the response time of the server processes. Therefore, the value of **LINK RANK** is not significant.

Display Format, FREEZE Attribute, Without DETAIL

Following is the format of the display returned by the STATUS SERVER FREEZE command without the DETAIL option. The fields are described in the following section:

STATUS SERVER FREEZE Display Format

```
SERVER           FREEZE STATE          TERM        FREEZE
 STATE
server-name       freeze-state       term-name freeze-state
    .                 .                 .          .
    .                 .                 .          .
    .                 .                 .          .
```

The fields in this display are as follows:

Display Field	Description	
SERVER	Name of the server class that has a freeze pending or is frozen	
FREEZE STATE	State of the server class. Possible values are:	
	FREEZE-PENDING	The server class received a FREEZE command, but the state of one or more terminals delays the freeze from completing.
	FROZEN	The server class is frozen. No communication can take place with any server of the server class until the class receives a THAW command.
TERM	Name of a terminal that prevents the server class from becoming frozen. This field is only displayed for server classes in a FREEZE-PENDING state (one entry for each terminal running under a local TCP that is preventing the freeze).	
FREEZE STATE	State of the server. This field is only displayed for server classes in a FREEZE-PENDING state (one entry for each terminal running under a local TCP that is preventing the freeze). Possible values are: FREEZE-PENDING, USING SERVER The terminal is currently communicating with the server class. FREEZE-PENDING, STOP-MODE NOT ZERO The value of the STOP-MODE special register is nonzero.	

Consideration

If the value of the MAXLINKS attribute for a server class is set too large, requests to a server class might be queued at the server. In this situation, the PATHMON process automatically keeps the link weight for the server artificially high to discourage the allocation of new links to the server class. You can resolve this situation by reconfiguring your Pathway environment to distribute the transaction load among more server processes.

Errors

This table lists the most common error that can occur during the processing of the STATUS SERVER command:

This Message…	Is Displayed When…
ERROR - *1045* REQUEST NOT VALID FOR CURRENT STATE	You have issued a `STATUS SERVER` command with the `FREEZE` option, but none of the server classes listed are in the FROZEN or FREEZEPENDING state.

Examples

This command displays detailed status information about all server classes:

```
STATUS SERVER *, DETAIL
```

This commands display status information about the specified frozen server classes, and include various options:

```
STATUS /OUT STATFLE/SERVER CLASS-1,FREEZE
STATUS SERVER CLASS-2,FREEZE
STATUS CLASS-3,FREEZE
```

This command displays the names of terminals that have not yet accepted the freeze condition for all server classes:

```
STATUS SERVER *, FREEZE
```

This command displays the names of all server classes that are in a FROZEN or FREEZE-PENDING state:

```
STATUS FREEZE
```

STOP SERVER Command

Use the `STOP SERVER` command to make all link manager that is the ACS subsystem processes and TCPs, return their links to a server class. The server class must be frozen before you issue this command.

```
STOP { [ SERVER ] server-class [ , scwait | forced] }
     {[ SERVER ] ( server-class [ , server-class ]... ) [ , scwait |
forced ] }
     {SERVER * [ , scwait | forced ] }
```

`server-class`

specifies the name of a previously defined and added server class.

`scwait`

< scwait> ::= WAIT < num> { HRS | MINS | SECS }

permits you to obtain original functionality of the `STOP` command after attempting the new style OPEN/CLOSE method and waiting for the specified < scwait> time.

NOTE: If the `<scwait>` option from PATHCOM is specified with `STOP SERVER` command, it will try to stop the server even if the server is in hung state. PATHCOM will wait for the specified <scwait> time and will return a command prompt after timeout.

WAIT number

specifies the time after which the never-linked, non associative server processes of the specified server class are to be stopped. This is meant for servers, which do not follow the Guardian OPEN/CLOSE logic.

SERVER *

stops all servers in all server classes that are in the FROZEN state. The PATHMON process does not report on servers it cannot stop.

forced

stops all the server processes (including associative server process started by this PATHMON) forcefully without waiting for the them to perform clean up or terminate themselves. This option can be used only when `scwait` option is not used.

Considerations

- A `STOP SERVER` command completes with a REQUEST PENDING message when all link managers have not yet returned their links to a stopped or abended server, or when a server that no longer has links to a link manager has not stopped.
- Program a server to stop (by calling the Guardian STOP procedure) after receiving a CLOSE message from a link manager. However, the never-linked, non-associative servers that do not follow Guardian OPEN/CLOSE logic, will continue to run. If the `WAIT` option is specified, the PATHMON process stops all the never-linked, non associative servers, using Guardian call Process_Stop_, after the specified WAIT time. PATHMON cannot distinguish between the servers that follow Guardian OPEN/CLOSE logic and that do not follow it. So, if WAIT time is specified for never-linked, non associative servers that follow Guardian OPEN/CLOSE logic, the WAIT time allows the servers to do the cleanup. If WAIT time is 0 secs, all the servers stop immediately. In case of duplicate `STOP SERVER` commands specified for the same server class with a different WAIT time, the never-linked, non associative servers of the specified server-class, will stop at the earliest timeout among the specified WAIT values.
- If an OSS server process starts another process not under the PATHMON process's control, the second process is not stopped when the PATHMON process stops the server class. For example, if an OSS server process runs another process, the executed process is outside the Pathway environment. When the PATHMON process stops the OSS server class, the executed process might become an orphan process.
- For information on stopping associative servers, see **Maintaining Associative Server Processes** on page 108.
- Use the `forced` option only after a normal STOP SERVER command is issued. Issue a STATUS command to verify that the server process(es) is in pending state before STOP SERVER command is issued with `forced` option.

Errors

This table lists the most common error that can occur during the processing of the `STOP SERVER` command:

This Message…	Is Displayed When…
1059 FREEZE PENDING	You have tried to issue a `STOP SERVER` command while a `FREEZE SERVER` command is still in progress; the `STOP SERVER` command fails.
	To stop the server, wait until the freeze has completed, then reissue the `STOP SERVER` command.

Examples

This command stops all the server classes in the Pathway environment. The never linked, non-associative servers not following Guardian OPEN/CLOSE logic, will stop after the specified WAIT time.

```
STOP SERVER *, WAIT 30 SECS
```

These commands stop the named server classes:

```
STOP CLASS-1
STOP SERVER CLASS-2
STOP (CLASS-3, CLASS-4)
```

This command stops all the server classes in the Pathway environment except neverlinked, non associative servers, that do not follow Guardian OPEN/CLOSE logic.

```
STOP SERVER *
```

THAW SERVER Command

Use the `THAW SERVER` command to a link manager that is the ACS subsystem processes and TCPs to resume sending requests to server processes in a frozen server class.

```
THAW { [ SERVER ] server-class }
     { [ SERVER ] ( server-class [, server-class ]... ) }
     { SERVER * }
```

server-class

specifies the name of a previously defined and added server class.

SERVER *

resumes communication with all server classes in the PATHMON configuration file.

Considerations

- After the PATHMON process thaws a server class, the PATHMON process allows Pathsend requestors access to the thawed server classes. For SCREEN COBOL requestors without an ON ERROR clause, the TCP resumes the operation by executing the send request. (When SCREEN COBOL requestors containing an ON ERROR clause encounter a frozen server class, they perform as directed by the ON ERROR code.)
- For an application running under Pathway/iTS, the TCP performs these after the `THAW` command resumes the activity of a terminal that was in TMF transaction mode:

  1. Directs the TMF software to restart the transaction at the BEGINTRANSACTION statement and to assign a new transaction identifier.
  2. Increments the special register RESTART-COUNTER by 1.

- Unlike the TCP, the ACS subsystem processes do not call any TMF procedures or increment any special TMF registers. It is the responsibility of the Pathsend requestors to detect any errors (by an error code) and proceed as required by the application.

Examples

These commands restart communication with the specified frozen server classes:

```
THAW SERVER CLASS-1
THAW (CLASS-2, CLASS-3)
```

This command resumes communication with all frozen server classes in the Pathway environment:

```
THAW SERVER *
```

Wild Card support in PATHCOM

The following SERVER commands in PATHCOM allow management and operation of the SERVER objects configured under a PATHMON, by specifying "*" in the command or in the SERVER object name:

- FREEZE
- INFO
- START
- STATUS
- STATS
- STOP
- THAW

NOTE:

1. Wild card support using only * in the command is applicable to TS/MP PATHCOM. (Refer **Wild card support using only * in the command** on page 243).
2. Wild card support by specifying "*" in the SERVER object name is applicable to TS/MP 2.6 or later PATHCOM.

For more information on TS/MP 2.6 or later, see the *TS/MP 2.7 ACS Reference Manual*.

Wild card support using only * in the command on page 243 shows the usage of wild card using only * in the command.

Wild card support using only * in the command

```
STATUS SERVER *
```

The above command displays status information for all server classes configured under PATHMON.

Wild card support using * in the SERVER object name on page 243 shows the usage of wild card using * in the SERVER object name.

Wild card support using * in the SERVER object name

```
STATUS SERVER EXM*
```

The above command displays the status of all server classes defined for the PATHWAY environment whose name starts with "EXM":

```
SERVER              #RUNNING            ERROR               INFO

EXMP1               1

EXMP2               1

EXMP3               1
```

Table Continued

```
EXMP4               1

EXMPA               1

EXMPB               1
```

NOTE: Only one instance of the wildcard character ("*") is accepted. If two or more asterisks are used, "illegal syntax" error appears. Therefore, for example, A*BD, AB*D or ABD* is supported. However, A*B*D or AB**D is not supported.

Usage Considerations for SET SERVER TIMEOUT Attribute

If a server process is delayed in its request processing (for example: when it is stuck behind a lock, is blocked by a higher priority process, or due to any performance pitfall), TS/MP might time out the request using the timeout value from either the SET SERVER TIMEOUT configuration parameter or the timeout argument of a SERVERCLASS_SEND_ call, SERVERCLASS_DIALOG_BEGIN_call or SERVERCLASS_DIALOG_SEND_call.

Despite being timed out, the server might continue to process the request or continue to be "stuck". Read-only type processing is especially vulnerable to these conditions.

When insert, update, or delete operations are being performed within a TMF transaction and a timeout occurs, the TMF transaction will be aborted. Subsequent I/O operations might then complete with file system error FEINVTRANSID, thereby terminating processing. Read-only processing has no easy way to know that the server request was cancelled, unless the server is multi-threaded and monitors $RECEIVE for incoming messages and cancellations while processing its current requests.

If MAXLINKS > 1, any requests queued on $RECEIVE will continue to queue there, possibly up to the point of timing out themselves. Configuring MAXLINKS 1 for a server class minimizes these additional effects.

Therefore, a single-threaded server is incapable of processing $RECEIVE “message cancellation” system messages in a timely fashion.

PATHMON Messages (Numbers 1000-1499)

This section describes the messages generated by the PATHMON process.

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. Message numbers identify the process that reported the error, as follows:

Message Numbers in This Range…	Are Generated by…
1000 through 1499	PATHMON process
1500 through 1999	PDMCOM process
2000 through 2999	PATHCOM process
3000 through 3999	Link managers (TCPs or ACS subsystem processes)

Messages 1000 through 1499 are described in this section. Messages 1500 through 1999 are discussed in **PDMCOM Messages (Numbers 1500-1999)** on page 281. Messages 2000 through 2999 are discussed in **PATHCOM Messages (Numbers 2000-2999)** on page 289. Messages 3000 through 3999 are described in **Link Manager Messages (Numbers 3000-3999)** on page 303. Some of these messages apply only to objects managed by the Pathway/iTS product, such as TCP, TERM, and PROGRAM objects. For information about those objects, see the *Pathway/iTS System Management Manual*.

This manual also includes log messages generated by LINKMON processes, described in **LINKMON Log Messages** on page 306.

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 **`inst-code`**	The instruction code of the program unit.
AT OFFSET **`%inst-addr`**	The instruction address of the program unit.
IN PROGRAM UNIT **`pu-name (pu-vrsn)`**	The name of the program unit followed by the version of the program unit in parentheses.
IN TCLPROG **`pu-file`**	The file where the program unit is located.

For example:

```
ERROR - *1048* TERM TERM-1, SUSPENDED: *3117* SERVER CLASS
```

```
UNDEFINED --- 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.

These PATHMON process errors include additional information:

1047 TERM STOPPED *1048* TERM SUSPENDED *1049* TERM SUSPENDED

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 by using the information obtained from the `STATUS TERM, DETAIL` command and the compiler listing with a MAP

PATHMON Messages

The following messages are returned by the PATHMON process:

1002

```
*1002* TERM term-name, ABORTED
```

Cause. The specified TERM object is aborted.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

1003

```
*1003* object, ALREADY STARTED
```

Cause. The named entity is already running.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1004

```
*1004* ALREADY STOPPED
```

Cause. The named entity is already stopped.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1005

```
*1005* PATHMON, SHUTDOWN
```

Cause. The PATHMON environment is successfully shut down.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1006

```
*1006* PATHMON, ABENDED
```

Cause. The PATHMON process aborted due to a fatal error.

Effect. The PATHMON process cannot process requests or monitor or control the PATHMON environment.

Recovery. Determine why the PATHMON process abended by checking the NonStop Kernel errors in messages logged just before this message, and then restart the PATHMON environment. If you cannot determine the cause of the error, contact your Hewlett Packard Enterprise representative, and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages 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), then supply your system number and the numbers and versions of all related products as well.

1007

```
*1007* link manager-name, CAN'T GRANT LINK : < nested-message
>
```

Cause. A server class link cannot be granted to a LINKMON process or a TCP.

Effect. The Pathsend statement or SEND operation fails.

Recovery. For additional information, see the description associated with the nested message.

1008

```
*1008* PATHCTL FILE CONTAINS NO VALID STATE
```

Cause. The PATHMON configuration file does not exist, or you used the `!` option when performing a cool start.

Effect. The PATHMON environment does not start.

Recovery. Re-enter your PATHMON environment configuration, assign a valid configuration file, or perform a cold start.

1009

```
*1009* PATHWAY CONFIGURATION CAN'T BE CHANGED
```

Cause. The `SET PATHWAY` command cannot be run while the PATHMON process is in the RUNNING state.

Effect. The operation fails.

Recovery. Stop the PATHMON environment before changing the configuration.

1010

```
*1010* PATHWAY NOT CONFIGURED
```

Cause. An attempt was made to run a command while the PATHMON process was not in the RUNNING state. Only the SET PATHMON and `SET PATHWAY` commands can be run while the PATHMON process is not running.

Effect. The command fails.

Recovery. Configure the PATHMON environment.

1011

```
*1011* link manager-name, DELINK FAILURE : < nested-message >
```

Cause. An error occurred while a TCP or LINKMON process was trying to return a link to a server class. The nested message indicates the reason for the failure.

Effect. The effect depends on the reason for the failure.

Recovery. For additional information, see the description associated with the nested message.

1012

```
*1012* object-name, ENTRY ALREADY EXISTS
```

Cause. Either the PATHMON process tried to add an object having a name that already exists, or the PATHMON process tried to start a TCP or server process using a process name that is already active within this PATHMON environment.

This message usually indicates an attempt to use an object name or a process name twice. For example, if this is a nested message in a CAN'T GRANT LINK error (for more information, see the description of error 1007), then the process name for a dynamic server already exists in the list of active server processes and TCPs for the PATHMON process.

Effect. The object is not added, or the TCP or server process is not started.

Recovery. Check the object name or the process name associated with the specified object. Change the name, as needed.

1013

```
*1013* object-name, EMPTY ENTRY
```

Cause. An entry that was supposed to contain the description of an object (for example, a server) is empty. This is an internal error.

Effect. The operation that needed to use the entry cannot complete.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1014

```
*1014* ENTRY IN USE (LOCKED)
```

Cause. The named object entry was locked (for example, during a state change) and the request timed out before the entry was unlocked.

Effect. The request fails.

Recovery. Reissue the request when the named object entry is no longer locked.

1015

```
*1015* object-name, ENTRY NOT DESIGNATED TYPE
```

Cause. An object type does not match the type specified or implied by a request.

Effect. The request that specified the object fails.

Recovery. Correct the object type specified or implied by the request, and retry the request.

1016

```
*1016* TCP tcp-name, FAILED
```

Cause. A TCP stopped abnormally even though a stop command was not issued.

Effect. The TCP is unavailable.

Recovery. Determine the cause of failure, and then restart the TCP.

1017

```
*1017* PATHCTL FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to the PATHMON configuration file.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1018

```
*1018* SERVER server-name, SERVER FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to the specified server class.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1019

```
*1019* link manager-name, PATHMON -> TCP FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to the specified TCP or LINKMON process.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1020

```
*1020* PATHMON, LOG1 FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to the LOG1 file.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1021

```
*1021* PATHMON, LOG2 FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to the LOG2 file.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1022

```
*1022* PATHMON, RECEIVE FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to $RECEIVE.

Effect. The effect depends on the file-system error that occurred.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1023

```
*1023* link manager-name, TCP -> PATHMON FILE ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred during an I/O operation to the specified TCP or LINKMON process.

Effect. The I/O operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1024

```
*1024* object-name, ILLEGAL FILE NAME (errnum)
```

Cause. An invalid file name was detected in the definition of the specified object. The **errnum** identifies the file-system error number.

Effect. The request that needed the object fails.

Recovery. Make sure that all Open Systems Services (OSS) pathnames are on the same system node as the PATHMON process; OSS processes cannot be spawned on a remote system. Correct all invalid file names in the definition of **object-name** .

1025

```
*1025* PATHMON, INTERNAL ERROR (%p-register)
```

Cause. An internal consistency check failed in the PATHMON process. This error is not fatal.

Effect. The PATHMON process creates a memory dump if the DUMP attribute is set to ON.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1026

```
*1026* PATHMON, ILLEGAL MESSAGE RECEIVED (USERID user-id)
```

Cause. A message received does not conform to the PATHMON request protocol. This error can be caused by running incompatible versions of PATHCOM, the PATHMON process, or PATHTCP. The message was received from the process running under the specified user ID.

Effect. The PATHMON process rejects the request.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1027

```
*1027* PATHMON, LOG1 FILE CLOSED
```

Cause. The LOG1 file is closed.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1028

```
*1028* PATHMON, LOG1 FILE OPENED
```

Cause. The LOG1 file is open.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1029

```
*1029* PATHMON, LOG2 FILE CLOSED
```

Cause. The LOG2 file is closed.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1030

```
*1030* PATHMON, LOG2 FILE OPENED
```

Cause. The LOG2 file is open.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1031

```
*1031* REQUIRED PATHWAY OR PATHMON CONFIGURATION PARAMETER
NOT SET
```

Cause. A required PATHMON or PATHWAY object attribute is not defined.

Effect. The operation fails.

Recovery. Define the required attribute. For details about the required attributes for those commands, see the `SET PATHMON` and the `SET PATHWAY` commands.

1032

```
*1032* PATHCTL FILE IS FULL
```

Cause. An entry cannot be added because the PATHMON configuration file is full. This is an internal error.

Effect. Entries cannot be added until the configuration file is available.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1033

```
*1033* PATHMON, NO INTERNAL FREESPACE AVAILABLE
```

Cause. The PATHMON process free-space table has insufficient space.

Effect. The operation fails.

Recovery. Contact your Hewlett Packard Enterprise representative for assistance.

1034

```
*1034* link manager-name, NO SERVER AVAILABLE FOR REQUESTING
TCP
```

Cause. The PATHMON process cannot grant a link to a TCP or a LINKMON process because no server processes are available. Either the TCP or LINKMON process is already linked to all server processes in the class, or all processes in the class currently have the maximum number of links (defined in the MAXLINKS attribute).

Effect. The requesting TCP or LINKMON process is not linked to the server class.

Recovery. Increase the value of the MAXSERVERS, MAXLINKS, or LINKDEPTH attribute in the server class definition. (Hewlett Packard Enterprise does not recommended setting LINKDEPTH higher than 1 for servers other than communications-based servers such as HLS servers.)

1035

```
*1035* object-name, NO SUCH ENTRY
```

Cause. The specified entry does not exist. Either the incorrect object name was used or the desired object does not exist.

Effect. The operation fails.

Recovery. Correct the object name or add the object to the PATHMON configuration file.

1036

```
*1036* object-name, NO SUCH SYSTEM
```

Cause. There is an invalid system name in the definition of the indicated object.

Effect. The operation fails.

Recovery. Use an ALTER command to correct the system name in the object definition.

1037

```
*1037* object-name, PARTIAL ENTRY DELETED (%internal-id)
```

Cause. An extraneous entry was deleted from the PATHMON configuration file during a cool start. These entry types are deleted:

- Temporary TERM entries that were created for a RUN PROGRAM request. This entry type only applies is you are running the Pathway/iTS product.
- Old copies of entries that were made obsolete as a result of an ALTER command
- Temporary server process entries that were named by the system

Effect. None

Recovery. Informational message only; no corrective action is needed.

1038

```
*1038* {SERVER server-name | TCP tcp-name}
PROCESS CREATION FAILURE: err-text
```

Cause. An error occurred during the creation of a new process for the indicated server or TCP. **err-text** identifies a process-creation error and provides detail concerning the problem.

Effect. The new process is not created.

Recovery. For additional information, see the description of the PROCESS_CREATE_ procedure in the *Guardian Procedure Calls Reference Manual*. For more detailed information including recovery actions, see the *Guardian Procedure Errors and Messages Manual*.

1039

```
*1039* PATHMON MUST BE NAMED
```

Cause. A PATHMON process name was not supplied when PATHMON was started.

Effect. The PATHMON process cannot be opened; the operation fails.

Recovery. Name your PATHMON process.

1040

```
*1040* UNABLE TO DETERMINE DEVICE TYPE (errnum)
```

Cause. A file-system error (**errnum**) was returned on a call to DEVICEINFO2.

Effect. The call to DEVICEINFO2 and the request fails. This error is not logged, but is simply returned to the operator.

Recovery. For information regarding the specified error, see to the *Guardian Procedure Errors and Messages Manual*.

1041

```
*1041* PATHMON, REQUEST HANDLER : < nested-message >
```

Cause. An error occurred during request initiation.

Effect. The request fails.

Recovery. For additional information, see the description associated with the nested message.

1042

```
*1042* TERM term-name, RESUMED
```

Cause. The indicated terminal has resumed processing.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1043

```
*1043* object-name, STARTED
```

Cause. The specified object has started.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1045

```
*1045* REQUEST NOT VALID FOR CURRENT STATE
```

Cause. The command cannot be executed because the object specified in the command or its controlling object (for example, the TCP for a TERM) is not in the appropriate state.

Effect. The request fails.

Recovery. Reissue the command when the indicated object or the controlling object is in the appropriate state.

1046

```
*1046* object-name, FILE-SYSTEM ERROR (errnum)
```

Cause. A file-system error (**errnum**) occurred.

Effect. The effect depends on the file-system error that occurred.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

1047

```
*1047* object-name, STOPPED : < nested-message >
```

Cause. The specified object has stopped.

Effect. The effect depends on the application and what type of object stopped.

Recovery. For additional information, see the description associated with the nested message.

1048

```
*1048* TERM term-name, SUSPENDED : < nested-message >
```

Cause. The specified terminal suspended processing due to an error in the program. The reason for the error is indicated in the nested message.

Effect. The terminal remains suspended until it is aborted with the `ABORT TERM` command.

Recovery. For additional information, see the description associated with the nested message.

1049

```
*1049* TERM term-name, SUSPENDED : < nested-message >
```

Cause. The specified terminal suspended processing due to a `SUSPEND TERM` command.

Effect. The terminal remains suspended until you issue the `RESUME TERM` command.

Recovery. For additional information, see the description associated with the nested message.

1050

```
*1050* PATHMON, SYSMSG HANDLER : < nested-message >
```

Cause. An error occurred while the PATHMON process was processing a system message. The error is indicated by the nested-message.

Effect. The effect depends on the particular error that occurred.

Recovery. For additional information, see the description associated with the nested message

1051

```
*1051* TCP tcp-name, EXCEEDS THE LIMIT ON PATHMON OPENERS.
```

Cause. The internal limit on the total number of concurrent openers of the PATHMON process would be exceeded by opening this TCP for communication with the PATHMON process at this time.

Effect. The PATHMON process cannot communicate with the TCP or LINKMON process.

Recovery. Stop some other TCPs, PATHCOM processes, or SPI communications or wait until they stop before trying to start this TCP again.

1052

```
*1052* TCP NOT DEFINED
```

Cause. The TCP specified in a TERM definition or a PROGRAM definition is not configured. The TCP must be configured and started before the associated TERMs and PROGRAMs are started.

Effect. The operation fails.

Recovery. Configure the TCP. To determine the name of the unconfigured TCP, use the `INFO PROGRAM` command.

1053

```
*1053* TCP NOT RUNNING
```

Cause. The TCP specified in a TERM definition is not running.

Effect. The operation fails.

Recovery. Determine why the TCP is not running. After correcting the problem, start the TCP with the `START TCP` command.

1054

```
*1054* link manager-name, TCP FAILED DURING REQUEST
```

Cause. The TCP or LINKMON process failed during a request.

Effect. The request is not completed.

Recovery. Determine the cause of the TCP or LINKMON process failure by checking the errors in messages logged just before this message.

After correcting the cause of the failure, restart the TCP (this step is not needed if AUTORESTART is specified for the TCP). A LINKMON process failure requires you to reload the processor.

If you cannot determine the cause of the error, contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1055

```
*1055* TOO MANY ENTRIES
```

Cause. The attempted request exceeds the number of concurrent executions allowed for the `RUN PROGRAM` command.

Effect. The PROGRAM terminates.

Recovery. Wait until a RUN PROGRAM request in progress completes and then initiate an unprocessed RUN PROGRAM request.

1057

```
*1057* ALREADY FROZEN
```

Cause. An attempt was made to freeze a server that is already frozen.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1058

```
*1058* FREEZE IN PROGRESS
```

Cause. A freeze request for the server is already in progress.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1059

```
*1059* FREEZE PENDING
```

Cause. A freeze request is accepted and will complete when all link managers pass a message to the PATHMON process that the server class is frozen.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1060

```
*1060* NOT FROZEN
```

Cause. An attempt was made to thaw a server that was not frozen or was already thawed.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1061

```
*1061* SERVER server-name, SERVER CLASS FROZEN
```

Cause. A TCP or ACS subsystem process link request was rejected because the specified server class is frozen. The server class must be thawed before links can be granted.

Effect. You cannot perform a SEND operation to a server class that is frozen.

Recovery. Thaw the server class, and then reissue the SEND request.

1062

```
*1062* MUST BE FROZEN
```

Cause. A `STOP SERVER` command was issued before the server class was frozen.

Effect. The command fails.

Recovery. Reissue the command after the server class is frozen.

1063

```
*1063* MUST BE THAWED
```

Cause. A `START SERVER` command was issued before the server class was thawed.

Effect. The command fails.

Recovery. Reissue the command after the server class is thawed.

1064

```
*1064* SERVER server-name, FROZEN
```

Cause. The specified server class is frozen.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1065

```
*1065* SERVER server-name, THAWED
```

Cause. The specified server class is thawed.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1066

```
*1066* TERMINALS NOT CONFIGURED TO RECEIVE TELL MESSAGES
```

Cause. You issued a `TELL` command but the SET PATHWAY MAXTELLQUEUE attribute value is set to 0.

Effect. The tell message request fails.

Recovery. Enter a value between 1 and 300 for the MAXTELLQUEUE attribute.

1067

```
*1067* INVALID INTERNAL IDENTIFIER FROM TCP
(%internal-identifier)
```

Cause. A TCP or ACS subsystem process request specified an entity that is not known to the PATHMON process. This is an internal error.

Effect. The request fails.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1068

```
*1068* PATHMON, MEMORY DUMP TAKEN file-name
```

Cause. A PATHMON process dump was taken by operator request or because the PATHMON DUMP attribute was d at the time an internal error occurred.

Effect. Memory is written to the named dump file.

Recovery. If the dump was taken by operator request, this message is informational only; no corrective action is needed. If an internal error occurred, contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1069

```
*1069* ERROR FROM SYSTEM PROCEDURE
```

Cause. The operating system returned an error on a call to a Guardian procedure.

Effect. The effect depends on the error.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1070

```
*1070* BACKUP PROCESS NOT UP
```

Cause. A `CONTROL`, `PRIMARY`, or `SWITCH` command cannot complete because the PATHMON process or the TCP backup process is not running.

Effect. The command fails.

Recovery. Start the object backup process, and reissue the command.

1071

```
*1071* PATHMON, CHECKPOINT FAILURE (%errnum)
```

Cause. An error occurred during an operating system CHECKPOINT procedure operation. **%errnum** is the CHECKPOINT error number.

Effect. The PATHMON process stops the backup process, if it is running, and attempts to start a new backup process.

Recovery. For information concerning the CHECKPOINT error, see the *Guardian Procedure Calls Reference Manual*.

1072

```
*1072* PATHMON, CREATE BACKUP FAILURE : < nested-message >
```

Cause. The PATHMON process could not create the backup process. The nested message indicates the reason.

Effect. The PATHMON process runs without a backup.

Recovery. For additional information, see the description associated with the nested message.

1073

```
*1073* PATHMON, CHECKOPEN FAILURE (%errnum)
```

Cause. An error occurred during an operating system CHECKOPEN operation. The error number identifies the CHECKOPEN error code.

Effect. The primary file is closed. Any additional effect depends on the file that was being opened.

Recovery. For information concerning the specified CHECKOPEN error, see the *Guardian Procedure Calls Reference Manual*.

1074

```
*1074* PATHMON, CHECKCLOSE FAILURE (%errnum)
```

Cause. An error occurred during an operating system CHECKCLOSE operation. The error number indicates the reason.

Effect. None

Recovery. If recovery is necessary, see the *Guardian Procedure Calls Reference Manual* for information concerning the specified CHECKCLOSE error.

1077

```
*1077* PATHMON, TAKEOVER BY BACKUP
```

Cause. The PATHMON backup process took over processing because the primary processor failed or because a `PRIMARY` or `SWITCH` command was run.

Effect. If the primary processor failed, the PATHMON process runs without a backup.

Recovery. If the primary processor failed, reload the processor so that the PATHMON process can create a backup process. If a `PRIMARY` or `SWITCH` command was run, this is an informational message only; no corrective action is needed.

1078

```
*1078* PATHMON, FATAL ERROR ( %P-register )
```

Cause. A PATHMON process internal-consistency check failed. This is an internal error.

Effect. The PATHMON process terminates.

If the DUMP attribute is d, the PATHMON process creates a dump file before terminating.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1079

```
*1079* NO STATIC SERVERS DEFINED
```

Cause. A `START SERVER` command failed because the SET SERVER NUMSTATIC attribute is set to 0.

Effect. The PATHMON process cannot start a server process until it receives a request from a the ACS subsystem process or a TCP.

Recovery. Define the SET SERVER NUMSTATIC attribute with a value from 1 through 4094.

1080

```
*1080* PATHMON, TRAP: T=%trap S=%s-reg P=%p-reg E=%e-reg L=%l-reg
```

Cause. A software or hardware failure resulted in a trap. This is an internal error. These values give the process environment at the time the PATHMON process failed:

%trap-no	is the number of the trap for use with DEBUG
%s-reg	is the stack register
%p-reg	is the program counter register
%e-reg	is the environment register
%l-reg	is the local register

Effect. The PATHMON process terminates. If possible, it creates a dump file.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1081

```
*1081* PATHCTL FILE IS INCOMPATIBLE (COLD START REQUIRED)
```

Cause. During a cool start, the PATHMON configuration file was not compatible with the currently running PATHMON process. This incompatibility can result from these either the PATHMON configuration file is a pre-D30 version of the file (which is not compatible with D30 versions of the PATHMON process), or the PATHMON configuration file has an invalid file code (the valid file code is 310).

Effect. The cool start fails.

Recovery. Cold start the PATHMON environment.

1083

```
*1083* link manager-name, LINE TO TCP IS DOWN
```

Cause. The Expand link to the external PATHMON environment is down.

Effect. The PATHMON process cannot access the specified LINKMON process or TCP.

Recovery. Establish a link to the system in which the LINKMON process or TCP is running.

1085

```
*1085* PATHMON MUST NOT RUN WITH HIGHPIN SET
```

Cause. You attempted to set the HIGHPIN attribute to ON for the PATHMON process.

Effect. The PATHMON process cannot run.

Recovery. Reset the HIGHPIN attribute to OFF and reissue the `START` command.

1086

```
*1086* TERM TYPE NOT DEFINED FOR PROGRAM
```

Cause. The *TYPE* parameter of the `RUN PROGRAM` command is undefined or mismatched, or the PROGRAM definition has not been added.

Effect. The operation fails.

Recovery. Supply a value for the RUN PROGRAM TYPE parameter.

1087

```
*1087* TERMINAL STOPPED BY OPERATOR
```

Cause. The terminal running the program was stopped by the `STOP TERM` command.

Effect. The terminal operation stops.

Recovery. Informational message only; no corrective action is needed.

1088

```
*1088* TERMINAL ABORTED BY OPERATOR
```

Cause. The terminal running the program was aborted by the `ABORT TERM` command.

Effect. The terminal operation terminates.

Recovery. Informational message only; no corrective action is needed.

1089

```
*1089* TERMINAL ABORTED BECAUSE OF ERROR
```

Cause. The terminal running the program encountered an error that caused the terminal to suspend.

Effect. The terminal operation is aborted.

Recovery. Check the PATHMON process log file for the Pathway and Guardian error encountered.

1090

```
*1090* SECURITY VIOLATION
```

Cause. The requesting user failed the security check for the operation on the PATHMON-controlled object.

Effect. The user is prohibited from performing unauthorized actions on the object.

Recovery. Informational only; no corrective action is required.

1092

```
*1092* TERMINAL SUSPENDED BECAUSE OF ERROR
```

Cause. The terminal was suspended because of an error in the SCREEN COBOL program.

Effect. The terminal is left in the SUSPENDED state.

Recovery. Isolate and correct the problem within the SCREEN COBOL program. You can use the `STATUS TERM, DETAIL` command to obtain the name, version, and instruction address of the program unit executing when the error occurred. 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.

1093

```
*1093* object-name, BACKUP PROCESSOR DOWN
```

Cause. The backup processor for the PATHMON process or for the TCP is not running.

Effect. A backup process cannot be created.

Recovery. Reload the processor.

1094

```
*1094* PATHMON, BACKUP PROCESS FAILED
```

Cause. The PATHMON backup process abended or stopped unexpectedly.

Effect. The PATHMON process creates a backup process, if possible.

Recovery. If the problem persists, contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1095

```
*1095* ILLEGAL CPU NUMBER
```

Cause The processor number specified in an ALTER, SET, or CONTROL command was not in the range 0-15 or was not the number of a processor on your system.

Effect. The command fails.

Recovery. Reissue the command, supplying the correct processor number.

1100

```
*1100* TCP tcp-name, TOO MANY TCP ENTRIES
```

Cause. The number of TCP descriptions has reached the maximum value specified for the MAXTCPS attribute of the SET PATHWAY command.

Effect. The ADD TCP command fails.

Recovery. Shut down the PATHMON environment. Increase the value of the MAXTCPS attribute (assuming it is not at the maximum allowable value). Reissue the ADD TCP command.

1101

```
*1101* TERM term-name, TOO MANY TERM ENTRIES
```

Cause. The number of terminal descriptions has reached the maximum value specified for the MAXTERMS attribute or the MAXPROGRAMS attribute of the SET PATHWAY command.

Effect. The ADD TERM or RUN PROGRAM command fails.

Recovery. Increase the value of the MAXTERMS attribute and reissue the ADD TERM command; or, increase the value of the MAXPROGRAMS attribute and reissue the RUN PROGRAM command.

1102

```
*1102* SERVER server-class, TOO MANY SERVER ENTRIES
```

Cause. The number of server class descriptions has reached the maximum value specified for the MAXSERVERCLASSES attribute of the SET PATHWAY command.

Effect. The ADD SERVER or ALTER SERVER command fails.

Recovery. Increase the value of the MAXSERVERCLASSES attribute, and then reissue the ADD SERVER or ALTER SERVER command.

1103

```
*1103* SERVER server-class, TOO MANY SERVER PROCESS ENTRIES
```

Cause. The number of server process definitions has reached the maximum value specified for the MAXSERVERPROCESSES attribute of the SET PATHWAY command.

Effect. The ADD SERVER or ALTER SERVER command fails.

Recovery. Increase the value of the MAXSERVERPROCESSES attribute and reissue the ADD SERVER or ALTER SERVER command.

1104

```
*1104* SERVER server-class, TOO MANY STARTUP ENTRIES
```

Cause. The number of server startup definitions has reached the maximum value specified for the MAXSTARTUPS attribute of the SET PATHWAY command.

Effect. The ADD SERVER or ALTER SERVER command fails.

Recovery. Increase the value of the MAXSTARTUPS attribute and reissue the `ADD SERVER` or `ALTER SERVER` command.

1105

```
*1105* SERVER server-class, TOO MANY ASSIGN ENTRIES
```

Cause. The number of ASSIGN definitions has reached the maximum value specified for the MAXASSIGNS attribute of the `SET PATHWAY` command.

Effect. The `ADD SERVER` or `ALTER SERVER` command fails.

Recovery. Increase the value of the MAXASSIGNS attribute and reissue the `ADD SERVER` or `ALTER SERVER` command.

1106

```
*1106* SERVER server-class, TOO MANY PARAM ENTRIES
```

Cause. The number of PARAM definitions has reached the maximum specified for the MAXPARAMS attribute of the `SET PATHWAY` command.

Effect. The `ADD SERVER` or `ALTER SERVER` command fails.

Recovery. Increase the value of the MAXPARAMS attribute and reissue the `ADD SERVER` or `ALTER SERVER` command.

1107

```
*1107* TOO MANY TELL MESSAGE ENTRIES
```

Cause. The number of tell messages issued has reached the maximum specified for the MAXTELLS attribute of the `SET PATHWAY` command, or the value defined for the MAXTELLS attribute is too large.

Effect. The tell message requests fail.

Recovery. Increase or decrease the value of the MAXTELLS attribute, depending on the cause of the error.

1108

```
*1108* PROGRAM program-unit-name, TOO MANY PROGRAM ENTRIES
```

Cause. The number of PROGRAM descriptions has reached the maximum value specified for the MAXPROGRAMS attribute of the `SET PATHWAY` command.

Effect. The `ADD PROGRAM` command fails.

Recovery. Increase the value of the MAXPROGRAMS attribute, and then reissue the `ADD PROGRAM` command.

1109

```
*1109* TOO MANY DEFINE ENTRIES
```

Cause. An attempt was made to add a DEFINE definition to a server class, but the total number of DEFINEs for all server classes has reached the maximum value specified for the MAXDEFINES attribute.

Effect. The `ADD SERVER` or `ALTER SERVER` command fails.

Recovery. Increase the value of the MAXDEFINES attribute, and then reissue the `ADD SERVER` or `ALTER SERVER` command.

1112

```
*1112* REQUESTED FUNCTION NOT SUPPORTED BY THIS VERSION
```

Cause. The command issued is not supported by this version of NonStop TS/MP.

Effect. The command fails.

Recovery. Issue a command that is compatible with the currently running version of NonStop TS/MP.

1113

```
*1113* FUNCTION NOT ALLOWED ON AN EXTERNAL TCP
```

Cause. The command issued is not valid for an external TCP.

Effect. The command fails.

Recovery. Issue a command that is acceptable for use with external TCPs

1114

```
*1114* PATHMON, CP TABLE FULL - BACKUP BEING RE-INITIALIZED
```

Cause. The PATHMON process cannot send checkpoint information to the backup process due to an internal condition.

Effect. The backup is stopped and automatically recreated.

Recovery. Informational message only; no corrective action is needed.

1115

```
*1115* TCP tcp-name, TCP REFRESHED
```

Cause. The TCP has read the requested program unit code segment into memory after checking the SCREEN COBOL object directory file.

Effect. The TCP uses the latest version of each program unit.

Recovery. Informational message only; no corrective action is needed.

1116

```
*1116* CHECK-DIRECTORY ON FOR TCP; REFRESH-CODE IGNORED
```

Cause. A `REFRESH-CODE` command was entered, but the TCP is configured with the CHECK-DIRECTORY attribute set to ON. The TCP checks the SCREEN COBOL object program directory file at each CALL statement.

Effect. The `REFRESH-CODE` command fails.

Recovery. Set the CHECK-DIRECTORY attribute to OFF.

1117

```
*1117* TCP tcp-name, ALREADY PRIMARIED
```

Cause. The primary TCP was already running in the configured processor when the `PRIMARY TCP` command was issued.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1118

```
*1118* SELECTION NOT APPLICABLE
```

Cause. The selection criteria does not apply to the specified objects, because no association exists between the objects.

Effect. The command fails.

Recovery. Reissue the command, specifying objects that have a relationship with each other.

1119

```
*1119* TERM term-name, AUTO ABORTED
```

Cause. The specified suspended terminal automatically aborted before being restarted with AUTORESTART.

Effect. The terminal abends.

Recovery. Informational message only; no corrective action is needed.

1120

```
*1120* object-name, AUTORESTARTED
```

Cause. After a failure, AUTORESTART restarted the specified object.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1121

```
*1121* object-name, AUTORESTART DISABLED
```

Cause. The specified object reached the AUTORESTART limit configured for it.

Effect. The object is not restarted.

Recovery. Solve the problem that is causing the persistent failures, and then restart the object. See the operating system errors or messages logged just before this message to determine what caused the persistent failures. Once you start the object, AUTORESTART is re-d for the object.

1122

```
*1122* PATHMON, BACKUP PROCESS STOPPED
```

Cause. The PATHMON backup process was stopped by the primary process. This message normally occurs after the `CONTROL PATHMON, BACKUPCPU` command causes the backup process to change from one processor to another.

Effect. The PATHMON backup process is down. A new backup is created in the new processor as soon as possible.

Recovery. Informational message only; no corrective action is needed.

1123

```
*1123* PATHMON, MEMORY DUMP NOT TAKEN (internal-condition)
```

Cause. An attempt to perform a PATHMON memory dump failed; generally, a dump file already exists. This is an internal error.

Effect. No dump file is created.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1124

```
*1124* PATHMON, ALLOCATESEGMENT ERROR (errnum)
```

Cause. An error occurred, identified by the error number, on a call to the ALLOCATESEGMENT procedure.

Effect. The PATHMON process is unable to allocate an extended data segment. The PATHMON process terminates.

Recovery. For information concerning the specified ALLOCATESEGMENT error, see the *Guardian Procedure Errors and Messages Manual*.

1125

```
*1125* PATHMON, USESEGMENT ERROR (errnum)
```

Cause. An error occurred, identified by the error number, on a call to the USESEGMENT procedure.

Effect. The PATHMON process is unable to select a particular extended data segment. The PATHMON process terminates.

Recovery. For information concerning the specified USESEGMENT error, see the *Guardian Procedure Errors and Messages Manual*.

1126

```
*1126* PATHMON CANNOT EXECUTE ON THIS RELEASE OF THE NonStop
KERNEL
```

Cause. The PATHMON process uses the extended memory feature of the NonStop operating system. In addition, the PATHMON process uses features available in release C20 of the NonStop operating system. This message appears if the PATHMON process is run on a pre-C20 release of the NonStop operating system.

Effect. The PATHMON process abends after displaying this message.

Recovery. Use a C20 or later version of the NonStop operating system.

1130

```
*1130* SPI PROCESSING DISABLED
```

Cause. The Subsystem Programmatic Interface (SPI) facility is not available. The PATHMON process cannot handle requests from processes calling the SPI procedures.

Effect. The SPI requests fail.

Recovery. Use version C00 or later of the NonStop operating system and the PATHMON process.

1135

```
*1135* EMS PROCESSING DISABLED
```

Cause. The Event Management Service (EMS) facility is not available. The PATHMON process cannot handle requests from processes calling EMS procedures.

Effect. The requests fail.

Recovery. Use version C00 or later of the NonStop operating system and the PATHMON process.

1136

```
*1136* EMSINIT ERROR (%errnum)
```

Cause. An error occurred, identified by the error number, on a call to the EMSINIT procedure.

Effect. Any PATHMON logs open with EVENTFORMAT are closed as a result of this error.

Recovery. For information concerning the EMSINIT error, see the *Guardian Procedure Calls Reference Manual*.

1137

```
*1137* EMSADDSUBJECT ERROR (%errnum)
```

Cause. An error occurred, identified by the error number, on a call to the EMSADDSUBJECT procedure.

Effect. Any PATHMON logs open with EVENTFORMAT are closed as a result of this error.

Recovery. For information concerning the EMSADDSUBJECT error, see the *Guardian Procedure Calls Reference Manual*.

1138

```
*1138* EMSADDTOKENS ERROR (%errnum)
```

Cause. An error occurred, identified by the error number, on a call to the EMSADDTOKENS procedure.

Effect. Any PATHMON logs open with EVENTFORMAT are closed as a result of this error.

Recovery. For information concerning the EMSADDTOKENS error, see the *Guardian Procedure Calls Reference Manual*, and the *Guardian Procedure Errors and Messages Manual*.

1039

```
*1139* EVENTFORMAT NOT SUPPORTED FOR THIS DEVICE
```

Cause. An attempt was made to perform event message logging to a terminal, printer, or operator process created with a version of the NonStop operating system previous to the C00 version.

Effect. The event message logging fails.

Recovery. Send the event message to a device or process that supports event formatting.

1140

```
*1140* DEFINES SPECIFIED AT PATHMON CREATION ARE IGNORED
```

Cause. You started the PATHMON process specifying DEFINEs other than the _DEFAULTS DEFINE with the VOLUME attribute.

Effect. PATHMON ignores any DEFINEs other than the _DEFAULTS DEFINE with the VOLUME attribute.

Recovery. Specify a DEFINE message using the `SET SERVER DEFINE` command.

1141

```
*1141* SERVER server-name, DEFINE PROCEDURE ERROR
(DEFINE-errnum)
```

Cause. An error occurred, identified by the error number, in a procedure call to the DEFINE procedure library (for example, to DEFINERESTORE).

Effect. The operation fails.

Recovery. For information concerning the specified DEFINE error, see the *Guardian Procedure Errors and Messages Manual*.

1142

```
*1142* SAVED DEFINE LENGTH EXCEEDS PATHMON LIMIT
```

Cause. The size of the DEFINE definition being added to a server class exceeds the PATHMON internal limit.

Effect. The DEFINE definition is not added.

Recovery. Correct the size of the DEFINE.

1143

```
*1143* DEFINE NAME IN SAVED DEFINE IS INCORRECT
```

Cause. The name of the DEFINE definition being added to a server class caused an internal conflict within the PATHMON process.

Effect. The DEFINE definition is not added.

Recovery. Contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1144

```
*1144* SERVER server-name, DEFINE PROCEDURE DISABLED
```

Cause. The library procedure for a DEFINE (such as DEFINERESTORE) is unavailable.

Effect. The processing of DEFINE statements is disabled, making it impossible to add a DEFINE definition to a server class.

Recovery. Install a version of the NonStop operating system that supports DEFINESAVE and DEFINERESTORE—that is, version C10 or later.

1145

```
*1145* PROGRAM program-name, ILLEGAL TERM NAME term-name
```

Cause. A `RUN PROGRAM` command specified an invalid name for a terminal (for example, a name with more than seven characters).

Effect. The `RUN PROGRAM` command fails.

Recovery. Supply a valid name for the terminal, and then reissue the command.

1146

```
*1146* SERVER server-name, THAW NOT COMPLETED
```

Cause. A THAW SERVER request is in progress.

Effect. The request issued to the frozen server fails.

Recovery. Reissue the command after the server has successfully thawed.

1147

```
*1147* BACKUP IS INITIALIZED AND RUNNING
```

Cause. The PATHMON backup process is initialized, running, and ready to take over for the primary process, if necessary.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1148

```
*1148* MAXIMUM NUMBER OF WAITED RUN PROGRAM REQUESTS EXCEEDED
```

Cause. An attempt was made to initiate more than 100 waited RUN PROGRAM requests from PATHCOM and from SPI programs.

Effect. The additional requests are denied.

Recovery. Perform a nowait RUN PROGRAM request, or wait until a request in progress finishes and then reinitiate a request.

1149

```
*1149* INTERNAL TABLE OVERFLOW -- TOO MANY ENTRIES
```

Cause. There is insufficient space for additional entries in one of the PATHMON internal tables. This is an internal error.

Effect. The ASSIGN, DEFINE, PARAM, PROCESS, and STARTUP attributes cannot be configured for a server class. If you are running the Pathway/iTS product, additional PROGRAM, TELL, TERM, and TERM objects cannot be configured. No server process can be started whose name is not configured by the user.

Recovery. Contact your Hewlett Packard Enterprise representative, and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1150

```
*1150* STARTING SHUTDOWN - <mode>
```

Cause. The shutdown process has started in the indicated mode.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1151

```
*1151* PATHMON STOPPED
```

Cause. The `STOP PATHMON` command was issued.

Effect. None

Recovery. Informational message only; no corrective action is needed.

1152

```
*1152* SHUTDOWN ESCALATED
```

Cause. The shutdown process is escalated.

Effect. None

Recovery. Informational error message; no corrective action is needed.

1153

```
*1153* SHUTDOWN IN PROGRESS
```

Cause. The request just submitted cannot be serviced because a shutdown of the PATHMON environment is in progress.

Effect. None

Recovery. Informational error message; no corrective action is needed.

1154

```
*1154* SHUTDOWN REQUEST TIMED OUT
```

Cause. The shutdown operation has not completed, but the time specified by the TIMEOUT attribute has expired.

Effect. None

Recovery. Informational error message; no corrective action is needed.

1155

```
*1155* SHUTDOWN FAILED
```

Cause. An object in the PATHMON environment, such as a server class, could not be stopped.

Effect. The shutdown fails.

Recovery. Stop the object using the `TACL STOP` command and retry the shutdown.

1156

```
*1156* PATHCTL FILE NOW INACCESSIBLE DUE TO FILE SYSTEM
ERROR (errnum)
```

Cause. A file-system error occurred, identified by the error number, while the PATHMON process was trying to access the PATHMON configuration file. This message is entered in the log file, but is not returned to the user.

Effect. The command might fail, depending upon when the file-system error occurred during the processing of the command. If the command fails, error 1017 is returned to the user. In either case, the PATHMON configuration file cannot be accessed by subsequent commands.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*. Cool or cold start the PATHMON environment to make the PATHMON configuration file accessible again.

1157

```
*1157* PATHCTL FILE NOT ACCESSIBLE
```

Cause. A command is issued that accesses the PATHMON configuration file, but the file is not accessible due to a previously occurring file-system error. The cause of the file-system error is noted in message 1156 in the log. This message is not logged.

Effect. The command fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*. Cool or cold start the PATHMON objects to make the PATHMON configuration file accessible again.

1158

```
*1158* PATHMON UNINITIALIZED - DOUBLE FAILURE
```

Cause. The primary PATHMON process has failed and the backup PATHMON process has not been fully initialized.

Effect. The PATHMON process cannot process requests or monitor or control the PATHMON environment.

Recovery. Determine why the PATHMON process abended by checking the Guardian errors in messages logged just before this message, and then restart the PATHMON environment. If you cannot determine the cause of the error, contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages generated
- Your PATHMON configuration file and the 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.

1159

```
*1159* SERVER object-name, OSS PROCESS CREATION FAILURE:
(process-creation-detail)
```

Cause. An error occurred while the PATHMON process was trying to create the specified OSS server process.

Effect. The server process is not created.

Recovery. OSS server processes are created by calls to the Guardian PROCESS_SPAWN_ procedure. For additional information, see the detailed description in this message. For information about the PROCESS_SPAWN_ procedure call, see the *Guardian Procedure Calls Reference Manual*. For

information about PROCESS_SPAWN_ codes, see the *Guardian Procedure Errors and Messages Manual*.

1160

```
*1160* OSS NOT AVAILABLE ON THIS SYSTEM
```

Cause. This PATHMON environment does not support OSS server processes.

Effect. The configured OSS server processes cannot be started.

Recovery. Call your Hewlett Packard Enterprise representative to inquire about installing OSS support features on your system.

1161

```
*1161* SERVER CLASS DOES NOT MATCH REQUESTED PROCESSTYPE
```

Cause. You have issued an `INFO SERVER` command specifying a server class that is not valid for the PROCESSTYPE defined for that class.

Effect. The `INFO SERVER` command fails.

Recovery. Reissue the command for an appropriate server class.

1162

```
*1162* RESIZESEGMENT ERROR (errnum)
```

Cause. A error occurred during a call to the Guardian RESIZESEGEMENT or CHECKRESIZESEGMENT procedure.

Effect. The command fails.

Recovery. For additional information, see the descriptions of the RESIZESEGEMENT or CHECKRESIZESEGMENT procedures in the *Guardian Procedure Calls Reference Manual*.

1163

```
*1163* LICENSING ERROR: PATHWAY PRODUCTS MUST BE LICENSED
```

Cause. PATHMON could not start up because it could not confirm that either the NonStop TS/MP product or the Pathway /TS product is licensed for your site.

Effect. System startup fails.

Recovery. Either the NonStop TS/MP product or the Pathway/iTS product must be licensed before the PATHMON process can start an application. If you have secured a license for the NonStop TS/MP or Pathway/iTS product, perform the following steps:

1. Confirm that one or both licensing files–ZLICA58 (NonStop TS/MP) or ZLICSA59 (Pathway/iTS) are located in the $SYSTEM.ZPATHWAY subvolume.
2. Confirm that the files located in $SYSTEM.ZPATHWAY are secured N**x**N**x**, where **x** is any valid Guardian security setting.

If a licensing file is present and properly secured, applications running under the licensed product must start up and run as expected.

1164

```
*1164* LICENSING ERROR: (product-name) PRODUCT NOT LICENSED
```

Cause. You have issued a command for a NonStop product that is not licensed for your site. For example, you have tried to use PATHCOM to define a server class, but NonStop TS/MP is not licensed for your site.

Effect. The command fails.

Recovery. The product named in the error message, either NonStop TS/MP or Pathway/iTS, must be licensed before you can add objects controlled by that product. If you have secured a license for the product named in the message, perform the following steps:

1. Confirm that the appropriate licensing file—ZLICSA58 (NonStop TS/MP) or ZLICSA59 (Pathway/iTS)—is located in the $SYSTEM.ZPATHWAY subvolume.
2. Confirm that all files located in $SYSTEM.ZPATHWAY are secured N**x**N**x**, where **x** is any valid Guardian security setting.

If the appropriate licensing file is present and properly secured, you can define objects controlled by the licensed product.

1165

```
*1165* MAXIMUM NUMBER OF LINK MANAGERS EXCEEDED
```

Cause. The maximum number of ACS subsystem process, TCP, and external TCP processes has exceeded the total number of link managers allowed (currently 800).

Effect. The bad value (MAXEXTERNALTCPS, MAXLINKMONS or MAXTCPS) is rejected and the SET PATHWAY command fails.

Recovery. Re-enter the values of MAXEXTERNALTCPS, MAXLINKMONS and MAXTCPS, after ensuring that their sum does not exceed the value of the maximum link managers allowed (currently 800).

1166

```
*1166* AUDIT MSG <audit msg>
```

Cause. PATHCOM command (ABORT/ADD/ALTER/DELETE/FREEZE/STOP/THAW/START) is issued on the objects configured under PATHMON.

Effect. None.

Recovery. Informational message only; no corrective action is needed.

1167

```
*1167* TERM <Term-Name>, TERM SHOULD BE IN STOPPED STATE
```

Cause. You have issued a CONTROL TERM COUPLE request when the TERM is not in the STOPPED state.

Effect. The request is not completed successfully.

Recovery. Ensure that the TERM is in the STOPPED state before issuing the COUPLE request.

1168

```
*1168* REBALANCE PROCESSOR DOWN
```

Cause. The processor number mentioned in the `CONTROL PATHMON SPREBALANCECPU` command is down.

Effect. Rebalancing of the server processes will not be started.

Recovery. Reload the processor, and then reissue the command.

1169

```
*1169* NO SERVER PROCESS IS QUALIFIED FOR REBALANCING
```

Cause. No special server processes available for rebalancing.

Effect. None.

Recovery. Informational message only; no corrective action is required.

1170

```
*1170* INVALID MODE: PATHMON SPREBALANCE MODE NOT MANUAL
```

Cause. Pathway attribute SPREBALANCEMODE is not MANUAL.

Effect. Rebalancing of the server processes cannot be started.

Recovery. As the SPREBALANCEMODE is not MANUAL, this command is not required. If the SPREBALANCEMODE attribute is DISABLE, rebalancing cannot happen. If the SPREBALANCEMODE is AUTO, rebalancing happens automatically. Manual intervention is not required.

1171

```
*1171* REBALANCING STARTED
```

Cause. Rebalancing of the serverclass have started.

Effect. None.

Recovery. Informational message only. Corrective action is not required.

1172

```
*1172* REBALANCING COMPLETED
```

Cause. Rebalancing of the serverclass have completed.

Effect. None.

Recovery. Informational message only. Corrective action is not required.

1173

```
*1173* REBALANCING FAILED
```

Cause. Rebalancing of the serverclass have failed.

Effect. Operation fails.

Recovery. Issue `FREEZE` and `THAW` commands to the serverclass to rebalance the requests or reissue the `REBALANCE` command.

1178

```
*1178* SERVER server-name, DEFAULT LINK GRANT ALGORITHM IS USED
```

Cause. Default link grant algorithm is used to grant link.

Effect. None

Recovery. Informational message only; no corrective action is required.

1179

```
*1179* SERVER server-name, CAN'T RETRIEVE SYSTEM NAME
```

Cause. Failed to get system name from “NODENUMBER_TO_NODENAME_” GPC

Effect. None

Recovery. Informational message only; no corrective action is required.

PDMCOM Messages (Numbers 1500-1999)

This section describes the messages generated by the PDMCOM process.

General Information

Message numbers identify the originator of messages returned by the Pathway environment, as follows:

Message Numbers in This Range…	Are Generated by…
1000 through 1499	PATHMON process
1500 through 1999	PDMCOM process
2000 through 2999	PATHCOM process
3000 through 3999	Link managers (TCPs or ACS subsystem processes)

Messages 1000 through 1499 are described in **PATHMON Messages (Numbers 1000-1499)** on page 245. Messages 1500 through 1999 are discussed in this section. Messages 2000 through 2999 are discussed in **PATHCOM Messages (Numbers 2000-2999)** on page 289. Messages 3000 through 3999 are described in **Link Manager Messages (Numbers 3000-3999)** on page 303.

This manual also includes log messages generated by LINKMON processes, described in **LINKMON Log Messages** on page 306.

PDMCOM Messages

These messages are returned by the PDMCOM process:

1500

```
*1500* Illegal Syntax - SYNTAX error in command-name command
```

Cause. Illegal syntax entered for the command. The displayed error message will have more information about the syntax error.

Effect. The command fails.

Recovery. Correct the syntax and issue the command again.

1501

```
*1501* Failure starting process-name process
```

Cause. PDMCOM fails to launch the process indicated by **process-name** . The displayed error message will have more information about the syntax error.

Effect. The command fails.

Recovery. Correct the failure condition indicated in the error message and issue the command again.

1502

```
*1502* History buffer is empty
```

Cause. FC command was the first command to be issued at the PDMCOM prompt.

Effect. The command fails.

Recovery. Issue the FC command only if a previous command line exists.

1503

```
*1503* Command String Not Found
```

Cause. The string entered with the `FC` command does not match any prior command.

Effect. The command fails.

Recovery. Specify a valid string with the `FC` command and issue the command again.

1504

```
*1504* Invalid command number
```

Cause. `FC` command was issued with an invalid command number.

Effect. The command fails.

Recovery. Enter a valid command number, and then reissue the command.

1505

```
*1505* Help File PDMHELP is in invalid format
```

Cause. The PDMHELP help file is not a `101` coded file or is not in a readable format.

Effect. `HELP` command, issued to receive help on the `PDMCOM` commands or the PDMCOM error numbers, fails.

Recovery. Change the file code to `101`. If the file is not in a readable format, then reinstall the product.

1506

```
*1506* No help available for that topic
```

Cause. No help is available for the text entered after the help keyword. This error might be caused because the entered text is not a:

- `PDMCOM` command
- Keyword related to PDMCOM
- Valid error/warning number

Effect. The help on the requested topic does not appear.

Recovery. Enter a valid `PDMCOM` command, or a keyword related to PDMCOM, or a valid error/warning number, and issue the command again.

1507

```
*1507* Help File PDMHELP not found
```

Cause. The PDMHELP help file is not found at the location where PDMCOM resides.

Effect. `HELP` command, issued to receive help on the PDMCOM commands or the PDMCOM error numbers, fails.

Recovery. Place the PDMHELP help file at the location where PDMCOM resides. If the file is missing from the system, then reinstall the product.

1508

```
*1508* No help available
```

Cause. Help could not be made available at this time.

Effect. `HELP` command, issued to receive help on the PDMCOM commands or the PDMCOM error numbers, fails.

Recovery. Usually, this error follows error number 1507 or 1506. Correct the earlier error, and then reissue the command.

1509

```
*1509* FC String too long
```

Cause. The string entered with `FC` command is too long.

Effect. The command fails.

Recovery. This error is unlikely. Contact your Global Mission Critical Support Center (GMCSC) with a description of the application task that was in progress when the error was encountered.

1510

```
*1510* Invalid command for object type.
```

Cause. PDMCOM was not able to parse the command correctly.

Effect. The command fails.

Recovery. This error is unlikely. Contact your Global Mission Critical Support Center (GMCSC) with a description of the application task that was in progress when the error was encountered.

1511

```
*1511* COMMUNICATION FAILED WITH <process name>.
```

Cause. PDMCOM failed to communicate with the specified process name.

Effect. The command fails.

Recovery. Restart the Access Control Server (ACS). If this problem persists, contact your Hewlett Packard Enterprise NonStop representative.

1512

```
*1512* <freeze/thaw/stop/start> FAILED FOR SERVER <server
name> IN PATHMON <pathmon name>.
```

Cause. PDMCOM failed to FREEZE or STOP the specified server in the specified PATHMON.

Effect. The command is aborted.

Recovery. Determine the reason for failure and initiate the corrective action accordingly. If the error is displayed during execution of an `ALTER-DOMAIN` command, then check the configuration of the server throughout the domain. Any inconsistency must be corrected manually.

1513

```
*1513* INTERNAL ERROR - <error message>.
```

Cause. Some unexpected error occurred.

Effect. The command is aborted.

Recovery. Take appropriate action depending on the error message.

1514

```
*1514* LOCK FAILED FOR SERVER <server name> IN ENVIRONMENT
<domain name or pathmon name>.
```

Cause. The server entry, for which a `FREEZE`, `STOP`, `ALTER`, `THAW`, `START`, or `ALTER-DOMAIN` command was issued, is already locked.

Effect. The command is aborted.

NOTE: The `Lock failed` error is also displayed when PDMCOM fails to communicate with the ACS subsystem. If the error appears, verify that the ACS subsystem is up and running.

Recovery. Wait for some time, and then reissue the command.

1515

```
*1515* DOMAIN NOT OPEN.
```

Cause. OPEN <`domain name`> is not issued prior to a domain-level command.

Effect. The domain-related commands cannot be issued.

Recovery. Reissue the failed command after issuing an `OPEN <domain name>` command.

1516

```
*1516* SERVERCLASS <servername> NOT CONFIGURED IN SUFFICIENT
PATHMONS IN THE DOMAIN <domain name>.
```

Cause. The number of PATHMONs in which the serverclass is configured is less than or equal to one, or the weight of the partitioned PATHMON is 100.

Effect. The command is aborted.

Recovery. Reissue the command after adding the serverclass to more than one PATHMON in the domain or modifying the weights of the PATHMONs in the Domain.

1517

```
*1517* NONEXISTENT DOMAIN.
```

Cause. The specified domain name does not exist.

Effect. The domain is not opened.

Recovery. Specify the correct domain name, and then reissue the command.

1518

```
*1518* NEW REQUEST QUEUING FAILED.
```

Cause. Internal request of the ALTER-DOMAIN command to queue Pathsend requests failed.

Effect. The command is aborted.

Recovery. Reissue the ALTER-DOMAIN command.

1519

```
*1519* NEW REQUEST PROCESSING FAILED.
```

Cause. Internal request of the ALTER-DOMAIN command to execute the queued Pathsend requests failed.

Effect. The command is aborted.

Recovery. Reissue the ALTER-DOMAIN command.

1520

```
*1520* SYSTEM NAME SPECIFIED IS NOT THE LOCAL SYSTEM NAME.
```

Cause. The system name specified in the OPEN or ALTER-DOMAIN command is not the local system name.

Effect. The command is aborted.

Recovery. Reissue the command after correcting the system name.

1521

```
*1521* PATHMON <pathmon name> NOT PART OF DOMAIN <domain
name>.
```

Cause. The PATHMON process specified with the PATHMON attribute of the ALTER-DOMAIN command is not a part of the domain opened by the earlier OPEN command.

Effect. The command is aborted.

Recovery. Specify the correct PATHMON name, and then reissue the command.

1522

```
*1522* SERVERCLASS <server name> NOT PART OF PATHMON <pathmon
name>.
```

Cause. The server class specified in the ALTER-DOMAIN command is not a part of the PATHMON specified in the command.

Effect. The command is aborted.

Recovery. Specify the correct server class name, and then reissue the command.

1523

```
*1523* DOMAIN ALREADY PARTITIONED FOR SERVER.
```

Cause. The `CONTROL DOMAIN PARTITION` command was issued for a domain-server combination, which was already partitioned.

Effect. The command is rejected.

Recovery. Issue the `CONTROL DOMAIN UNDO PARTITION` command, and then reissue the command.

1524

```
*1524* DOMAIN NOT PARTITIONED FOR SERVER.
```

Cause. The `CONTROL DOMAIN UNDO PARTITION` command was issued for a domain-server combination, which was not partitioned.

Effect. The command is rejected.

Recovery. Issue the `CONTROL DOMAIN PARTITION` command, and then reissue the command.

1525

```
*1525* MAX LIMIT REACHED.
```

Cause. The number of `CONTROL DOMAIN PARTITON` command reached the maximum value—128.

Effect. The command is rejected.

Recovery. Reduce the number of partition requests.

1526

```
*1526* ACS INFRASTRUCTURE IS SHUTTING DOWN.
```

Cause. The command is issued when the ACS Infrastructure is shutting down.

Effect. The command is rejected.

Recovery. Start the ACS Infrastructure, and then reissue the command.

1527

```
*1527* OPEN ERROR ON OBEY FILE, ERROR=<error no>.
```

Cause. PDMCOM failed to open an OBEY file. <error no> indicates the error during the OPEN operation.

Effect. The command fails.

Recovery. Check the error number, and try to resolve the error condition. If the problem persists, contact your Hewlett Packard Enterprise NonStop representative.

1528

```
*1528* READ ERROR ON OBEY FILE.
```

Cause. PDMCOM failed to read an OBEY file.

Effect. The command fails.

Recovery. The OBEY file might be corrupt. Create a new OBEY file and reissue the command. If the problem persists, contact your Hewlett Packard Enterprise NonStop representative.

1529

```
*1529* FC COMMAND IS INVALID IN AN OBEY FILE.
```

Cause. The `FC` command is issued in an OBEY file.

Effect. The command fails.

Recovery. Remove the `FC` command from the OBEY file and reissue the command.

1530

```
*1530* REDISPLAY COMMAND IS INVALID IN OBEY FILE.
```

Cause. The Redisplay command is issued in an OBEY file.

Effect. The command fails.

Recovery. Remove the Redisplay command from the OBEY file and reissue the command.

1531

```
*1531* RE-EXECUTE COMMAND IS INVALID IN OBEY FILE.
```

Cause. The Re-execute command is issued in an OBEY file.

Effect. The command fails.

Recovery. Remove the Re-execute command from the OBEY file and reissue the command.

1532

```
*1532* HISTORY COMMAND IS INVALID IN OBEY FILE.
```

Cause. The `HISTORY` command is issued in an OBEY file.

Effect. The command fails.

Recovery. Remove the `HISTORY` command from the OBEY file and reissue the command.

1533

```
*1533* CAN'T QUALIFY GIVEN FILE NAME WITH DEFAULT VOLUME
INFORMATION.
```

Cause. PDMCOM cannot interpret the file name because it might be invalid.

Effect. The command fails.

Recovery. Check the file name supplied. Correct any error in the file name. If the problem persists, contact your Hewlett Packard Enterprise NonStop representative.

1534

```
*1534* OBEY FILENAME IS MISSING.
```

Cause. The OBEY file name is not supplied.

Effect. The command fails.

Recovery. Specify an OBEY file name and reissue the command.

1535

```
*1535* NON-PRINTABLE CHARACTERS IN OBEY FILE.
```

Cause. The OBEY file supplied contains nonprintable characters.

Effect. The command fails.

Recovery. Check the OBEY file supplied. Remove any nonprintable characters from the OBEY file and reissue the command.

1536

```
*1536* DOMAIN or PATHMON NOT OPEN.
```

Cause. No OPEN is issued prior to a command.

Effect. The command fails.

Recovery. Reissue the failed command after issuing an OPEN <*domain name*> or OPEN <*pathmon name*> command.

1537

```
*1537* ILLEGAL NUMBER.
```

Cause. The number format entered is not valid.

Effect. The command fails.

Recovery. Supply a valid number.

1538

```
*1538* OPERATION NOT ALLOWED - <DETAIL>.
```

Cause. The command execution is restricted due to any of the following conditions:

- Serverclass that is rebalanced is partitioned.
- Serverclass is locked by another operation.
- `CONTROL ACS` command is in progress.
- Another `REBALANCE-DOMAIN` command is in progress.
- `DOMAINLLBA` option is not set to `WBLU`.

Effect. The command fails.

Recovery. Reissue the command after rectifying the condition(s).

PATHCOM Messages (Numbers 2000-2999)

This section lists the messages generated by PATHCOM.

General Information

Message numbers identify the originator of messages returned by the Pathway environment, as follows:

Message Numbers in This Range…	Are Generated by…
1000 through 1499	PATHMON process
1500 through 1999	PDMCOM process
2000 through 2999	PATHCOM process
3000 through 3999	Link managers (TCPs or ACS subsystem processes)

Messages 1000 through 1499 are described in **PATHMON Messages (Numbers 1000-1499)** on page 245. Messages 1500 through 1999 are discussed in **PDMCOM Messages (Numbers 1500-1999)** on page 281. Messages 2000 through 2999 are discussed in this section. Messages 3000 through 3999 are described in **Link Manager Messages (Numbers 3000-3999)** on page 303. Some of these messages apply only to objects managed by the Pathway/iTS product, such as TCP, TERM, and PROGRAM objects. For information about those objects, see the *Pathway/iTS System Management Manual*.

This manual also includes log messages generated by LINKMON processes, described in **LINKMON Log Messages** on page 306.

Some PATHCOM 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.

PATHCOM Messages

These messages are returned by PATHCOM:

2001

```
*2001* TERM term-name, ABORTED
```

Cause. The specified terminal was aborted.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2002

```
*2002* ABORT FAILURE
```

Cause. An abort request failed.

Effect. None

Recovery. Determine the cause for the failure and reissue the abort request.

2003

```
*2003* PATHMON NOT OPEN
```

Cause. The PATHMON process is not open.

Effect. The request to the PATHMON process fails.

Recovery. Use the `PATHCOM OPEN` command with the PATHMON process name.

2007

```
*2007* SERVER server-name FROZEN
```

Cause. The indicated server class is frozen.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2008

```
*2008* ILLEGAL ATTRIBUTE VALUE
```

Cause. The value specified for an attribute is not within the valid range.

Effect. The command fails.

Recovery. Respecify the value in the valid range for that attribute.

2009

```
*2009* ILLEGAL CPU NUMBER
```

Cause. The processor number is not in the range 0-15, or the same processor number is defined for the objects primary and backup processes.

Effect. The command fails.

Recovery. Redefine the processor number with a correct value.

2010

```
*2010* ILLEGAL FILENAME
```

Cause. A file name did not conform to file-naming conventions. PATHCOM checks for valid network file names in the ASSIGN, IN, OUT, PROGRAM, SWAP, TCLPROG, FILE, and PRINTER attributes.

Effect. The command fails.

Recovery. Rename the file with a valid name.

2011

```
*2011* ILLEGAL NUMBER
```

Cause. The number format entered is not valid.

Effect. The command fails.

Recovery. Enter a valid number.

2012

```
*2012* ILLEGAL PRIORITY VALUE {0:255}
```

Cause. The priority value entered is not in the 0-255 range.

Effect. The command fails.

Recovery. Enter a valid priority value.

2013

```
*2013* ILLEGAL SYNTAX
```

Cause. The syntax for the newly introduced or modified command is incorrect.

Effect. The command fails.

Recovery. Enter the input characters using the correct syntax.

2014

```
*2014* ILLEGAL TIME VALUE
```

Cause. The time value entered cannot be represented in 16 bits.

Effect. The command fails.

Recovery. Enter an acceptable time value.

2015

```
*2015* PATHCOM INTERNAL ERROR (%p-reg)
```

Cause. An internal consistency check failed in PATHCOM.

Effect. The contents of the program register (%**p-reg**) is displayed; PATHCOM does not terminate.

Recovery. Contact your Global Customer Support Center (GCSC).Before contacting the GCSC, have the following information available:

- Log files and dump files
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

This information is invaluable for the support staff to be able to understand the problem and help fix the problem.

2016

```
*2016* INVALID REPLY FROM PATHMON
```

Cause. A reply was received that does not comply with the NonStop TS/MP internal interprocess message protocol. This error can be caused by:

- Running incompatible versions of PATHCOM and the PATHMON process.
- PATHCOM attempting communication with a process that is not a PATHMON process.
- Running a program on terminals with non-network names. Terminal names used in a PATHMON environment cannot exceed seven characters including the $ symbol.

Effect. The reply fails.

Recovery. Depending on the cause, run compatible versions of PATHCOM and the PATHMON process, ensure that PATHCOM is accessing a PATHMON process, or provide network names for terminals.

2017

```
*2017* MISSING CLOSING QUOTE
```

Cause. The closing quotation mark is missing from a character string.

Effect. The command fails.

Recovery. Enter the closing quotation mark and resubmit the command.

2018

```
*2018* REQUIRED PARAMETER NOT SET
```

Cause. A required attribute is missing, or the attribute is invalid for this command.

Effect. The command fails.

Recovery. Enter the required attribute and resubmit the command.

2019

```
*2019* NO INTERNAL FREESPACE AVAILABLE
```

Cause. No free space is available in the PATHCOM internal data area.

Effect. The operation fails.

Recovery. Contact your Global Customer Support Center (GCSC).

Before contacting the GCSC, have the following information available:

- Log files and dump files
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

This information is invaluable for the support staff to be able to understand the problem and help fix the problem.

2020

```
*2020* NO SUCH ENTRY
```

Cause. The object name in a PATHCOM command does not exist.

Effect. The command fails.

Recovery. Enter a correct object name and resubmit the command.

2021

```
*2021* PATHCOM OPERATOR INPUT NEEDED IN NON-INTERACTIVE MODE
```

Cause. A response is needed but PATHCOM is running noninteractively. This message is equivalent to the interactive DO YOU WISH TO CONTINUE? prompt.

Effect. Because PATHCOM is in noninteractive mode, the process terminates.

Recovery. Enter the required response.

2022

```
*2022* TERM term-name RESUMED
```

Cause. The named terminal has resumed operation.

Effect. None

Recovery. Informational message only; no corrective action is needed.

NOTE: You will receive this message only if you are using the Pathway/iTS product.

2024

```
*2024* object-name STARTED
```

Cause. The named object is running.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2027

```
*2027* object-name STOPPED
```

Cause. The named object is stopped.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2028

```
*2028* STRING TOO LONG
```

Cause. An input string exceeded the maximum acceptable length.

Effect. The request fails.

Recovery. Resubmit the input string with a valid length.

2029

```
*2029* term-name SUSPENDED
```

Cause. The named terminal is suspended.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2031

```
*2031* TCP tcp-name SWITCHED
```

Cause. The primary and backup processes of the named TCP exchanged operation in response to a `TCP SWITCH` command.

Effect. The command completed successfully.

Recovery. Informational message only; no corrective action is needed.

NOTE: You will receive this message only if you are using the Pathway/iTS product.

2032

```
*2032* TABLE OVERFLOW
```

Cause. This is an internal PATHCOM error.

Effect. The operation fails.

Recovery. Contact your Global Customer Support Center (GCSC).

Before contacting the GCSC, have the following information available:

- Log files and dump files
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

This information is invaluable for the support staff to be able to understand the problem and help fix the problem.

2033

```
*2033* TELL PENDING, MSGID = number
```

Cause. A tell message with the indicated message identification is pending.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2034

```
*2034* SERVER server-name THAWED
```

Cause. The named server class is thawed.

Effect. None

Recovery. Informational message only; no corrective action is needed.

2035

```
*2035* TOO MANY NESTED OBEY FILES (MAX=4)
```

Cause. The number of nested command files exceeded the maximum depth of 4.

Effect. The command fails.

Recovery. Rewrite the command files with 4 or less nested files.

2036

```
*2036* TOO MANY CPU PAIRS SPECIFIED (MAX=16)
```

Cause. The number of processor pairs exceeded the maximum number of 16.

Effect. The command fails.

Recovery. Specify 16 or less processor pairs.

2037

```
*2037* NUMBER OF PREDEFINED SERVERS WOULD EXCEED MAXSERVERS
```

Cause. The number of defined servers exceeded the maximum value defined for the MAXSERVERS attribute.

Effect. The command fails.

Recovery. Reduce the number of defined servers or increase the value of the MAXSERVERS attribute in the `SET SERVER` command.

2038

```
*2038* TOO MANY PARAMS
```

Cause. The limit for the configured number of PARAMs is reached.

Effect. The `SET SERVER PARAM` command fails.

Recovery. Reduce the number of PARAM messages.

2040

```
*2040* PATHCOM TRAP T = %trap S = %s-reg P = %p-reg
E = %e-reg L = %l-reg
```

Cause. PATHCOM is trapped because of a hardware or software failure. These values reflect the process environment at the time PATHCOM failed:

%**trap**	is the number of the trap.
%s**-reg**	is the stack register.
%p**-reg**	is the program counter register.

Table Continued

%e**-reg**	is the environment register.
%l**-reg**	is the local register.

Effect. PATHCOM fails.

Recovery. Contact your Global Customer Support Center (GCSC).

Before contacting the GCSC, have the following information available:

- Log files and dump files
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

This information is invaluable for the support staff to be able to understand the problem and help fix the problem.

2041

```
*2041* PATHWAY MICROCODE NOT INSTALLED
```

Cause. The microcode required for operating NonStop TS/MP has not been installed in the system.

Effect. NonStop TS/MP cannot run.

Recovery. Load the required NonStop TS/MP microcode and start the PATHMON objects.

2042

```
*2042* THIS FORM OF PROCESS NAME NOT ALLOWED
```

Cause. The process name contained more than six characters including the dollar sign ($).

Effect. The command fails.

Recovery. Reenter the process name with the correct amount of characters.

2043

```
*2043* ILLEGAL SECURITY ATTRIBUTE
```

Cause. The value specified for a security attribute was invalid.

Effect. The command fails.

Recovery. Enter the correct security attribute.

2045

```
*2045* ILLEGAL USER NAME
```

Cause. The value specified for a user name was invalid.

Effect. The command fails.

Recovery. Enter the correct user name.

2046

```
*2046* ILLEGAL LINKDEPTH VALUE {1:255}
```

Cause. The value specified for the LINKDEPTH attribute was either 0 or greater than 255.

Effect. The command fails.

Recovery. Enter a correct value for the LINKDEPTH attribute.

2047

```
*2047* RUN PERMITTED ONLY FOR IN = OUT = TERMINAL
```

Cause. A `RUN` command is allowed in interactive mode only.

Effect. The command fails.

Recovery. Re-enter the command in interactive mode.

2048

```
*2048* ILLEGAL TERM TYPE SPECIFICATION
```

Cause. The value specified for TERM TYPE is invalid.

Effect. The command fails.

Recovery. Enter a correct value for TERM TYPE.

NOTE: You will receive this message only if you are using the Pathway/iTS product.

2049

```
*2049* NUMSTATIC EXCEEDS MAXSERVERS
```

Cause. The number of static servers defined for a server class is greater than the maximum value specified for the MAXSERVERS attribute for that class.

Effect. The command fails.

Recovery. Reduce the number of static servers defined or increase the value of the MAXSERVERS attribute.

2050

```
*2050* ERROR NUMBER NOT DEFINED
```

Cause. There is no PATHMON environment error defined with this number.

Effect. The error request fails.

Recovery. Enter a valid error number.

2051

```
*2051* CODE REFRESHED
```

Cause. The TCP has read the requested program unit code segment into memory after checking the SCREEN COBOL object directory file.

Effect. The TCP uses the latest version of each program unit.

Recovery. Informational message only; no corrective action is needed.

2054

```
*2054* TCP tcp-name PRIMARIED
```

Cause. The primary process is running in the configured processor.

Effect. The `PRIMARY TCP` command completed successfully.

Recovery. Informational message only; no corrective action is needed.

2056

```
*2056* IN FILE CANNOT RUN NOWAIT
```

Cause. A `RUN PROGRAM` command was issued with the `NOWAIT` option but the FILE attribute was not specified; or `NOWAIT` was specified, but the name of the PATHCOM command terminal was entered for the FILE attribute. A program cannot be run with the `NOWAIT` option set on the PATHCOM command terminal.

Effect. The command fails.

Recovery. Either specify the FILE attribute or specify a terminal other than the PATHCOM command terminal when using the `NOWAIT` option.

2057

```
*2057* PATHCOM CANNOT EXECUTE ON THIS RELEASE OF THE
OPERATING SYSTEM
```

Cause. An attempt was made to execute PATHCOM on a version of the NonStop operating system previous to the C40 release. PATHCOM requires the features supported by later releases of the operating system.

Effect. PATHCOM abends.

Recovery. Obtain and install the correct release of the operating system. Contact your Global Customer Support Center (GCSC).

Before contacting the GCSC, have the following information available:

- Log files and dump files
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

This information is invaluable for the support staff to be able to understand the problem and help fix the problem.

2058

```
*2058* RESET NOT ALLOWED FOR THIS PARAMETER
```

Cause. An `ALTER` command was issued that attempted to reset a required attribute.

Effect. The command fails.

Recovery. Stop the PATHMON environment, redefine the attribute using the `SET` command, and restart the PATHMON environment.

2059

```
*2059* DEFINE PROCEDURE ERROR (DEFINE-errnum)
```

Cause. An error occurred, identified by the error number, in a procedure call to the DEFINE procedure library.

Effect. The operation fails.

Recovery. For information concerning the specified DEFINE error, see the *Guardian Procedure Errors and Messages Manual*.

2060

```
*2060* SAVED DEFINE LENGTH EXCEEDS PATHCOM LIMIT
```

Cause. The size of the DEFINE definition being added to a server class exceeds the PATHMON process internal limit.

Effect. The DEFINE definition is not added.

Recovery. Correct the size of the DEFINE.

2061

```
*2061* DEFINE PARSING AND DISPLAY DISABLED
```

Cause. PATHCOM is not able to parse or display the DEFINE definitions. This error can occur, for example, if one or more of the required DEFINE procedures, such as DEFINESAVE and DEFINERESTORE, are not available.

Effect. The DEFINE operation fails.

Recovery. Contact your Global Customer Support Center (GCSC).

Before contacting the GCSC, have the following information available:

- Log files and dump files
- Your PATHMON configuration file and PATHMON process version
- A description of the application task that was in progress when the error was encountered

This information is invaluable for the support staff to be able to understand the problem and help fix the problem.

2063

```
*2063* SAVED CONFIGURATION IN PATHCTL FILE IS VALID FOR COOL
START. DO YOU WISH TO OVERWRITE IT?
```

Cause. The `START PATHWAY COLD` command (without the `!` option) was issued, the PATHCTL file exists, and the stored configuration is valid.

Effect. The PATHMON environment can be cool started from the existing PATHCTL file.

Recovery. If you choose to continue the command, the current PATHCTL file is overwritten.

2064

```
*2064* CURRENT PATHMON NAME <process-name> IS DIFFERENT FROM
PATHMON NAME <process-name> IN PATHCTL FILE. DO YOU WISH TO
CONTINUE?
```

Cause. The `START COLD` command (without the `!` option) was issued. Although the PATHCTL file exists and the stored configuration is valid, the PATHMON process name specified in the `START` command is different from the PATHMON process name stored in the PATHCTL file.

Effect. The PATHMON environment can be cool started from the existing PATHCTL file using the old PATHMON process name, or the PATHMON environment can be cold started using the new PATHMON process name specified in the `START` command.

Recovery. If you choose to continue, the current PATHCTL file is overwritten and the new PATHMON process name is used.

2066

```
*2066* INVALID FIELD: GUARDIAN ATTRIBUTE SET FOR OSS SERVER
```

Cause. An attribute you defined for an OSS server process is valid only for Guardian server processes.

Effect. The `SET`, `ADD`, or `ALTER` command fails.

Recovery. Remove the inappropriate attribute definition from the server class configuration by using the `RESET SERVER` command.

2067

```
*2067* INVALID FIELD: OSS ATTRIBUTE SET FOR GUARDIAN SERVER
```

Cause. An attribute you defined for a Guardian server process is valid only for OSS server processes.

Effect. The `SET`, `ADD`, or `ALTER` commands fail.

Recovery. Remove the inappropriate attribute definition from the server class configuration by using the `RESET SERVER` command.

2068

```
*2068* AN OSS ATTRIBUTE HAS A LENGTH GREATER THAN IS ALLOWED
```

Cause. The value for an OSS server process attribute exceeded the maximum length.

Effect. The `SET`, `ADD`, or `ALTER` command fails.

Recovery. Define a value for the attribute within the valid range.

2069

```
*2069* OSS PATHNAME IS INVALID
```

Cause. An OSS pathname in the command is invalid due to a disallowed forward slash (/) or null character.

Effect. The attribute value for the OSS server process is not set.

Recovery. Make sure that OSS pathnames (other than those specified for the CWD attribute or the `CMDCWD` command) do not have a forward slash (/) as the last character.

2070

```
*2070* THE COMBINED LENGTH OF ARGLIST AND ENV IS GREATER
THAN IS ALLOWED
```

Cause. The combined length of the values defined for the server class attributes ARGLIST and ENV exceed the 24,000-character maximum.

Effect. Whichever attribute was specified last, ARGLIST or ENV, is not set.

Recovery. Set the ARGLIST and ENV values within the allowed maximum length. See the description of the ARGLIST and ENV attributes in **Managing the Pathsend Environment** on page 112.

2071

```
*2071* OSS PATHNAME RESOLVED WITH CWD EXCEEDS MAXIMUM
LENGTH: attribute-name
```

Cause. A relative pathname was entered for the specified OSS attribute. When PATHCOM resolved the relative pathname, using the OSS directory specified in the CWD attribute or the `CMDCWD` command, the resulting absolute pathname exceeded the maximum 1023 bytes allowed.

Effect. The server class `ADD` or `ALTER` command fails.

Recovery. Redefine the default directory or the relative pathname for the specified attribute within the allowed maximum length.

2072

```
*2072* OSS PATHNAME NOT ABSOLUTE OR DOESN'T RESOLVE TO
ABSOLUTE PATHNAME: attribute-name
```

Cause. The value specified for the CWD attribute was not an absolute pathname. If no pathname is defined for the CWD attribute, and then the pathname entered for the specified OSS attribute is not an absolute pathname.

Effect. If you attempted to set the CWD attribute, the pathname is rejected and the attribute is not set. If you attempted to add or alter other OSS attributes, the `ADD` or `ALTER` command fails.

Recovery. Either specify an absolute pathname for the given attribute or set an absolute pathname using the `SET SERVER CWD` command or the `CMDCWD` command. See the descriptions of the OSS attributes in **Managing the Pathsend Environment** on page 112.

2073

```
*2073* PROCESSTYPE DOESN'T MATCH ATTRIBUTES CURRENTLY SET
```

Cause. The value specified for the PROCESSTYPE attribute conflicts with values set for other attributes in the server class.

Effect. The value for the PROCESSTYPE attribute is not changed.

Recovery. Reset all conflicting attribute values before changing the value of PROCESSTYPE.

2075

```
*2075* CMDVOL NODE VALUE CONFLICTS WITH PATHWAY
NODEINDEPENDENT ON
```

Cause. The SET PATHWAY NODEINDEPENDENT attribute is set to ON, but a value other than * was specified by the `CMDVOL` command as the node name in the default settings for file name expansion.

Effect. The * node name specification conflicts with the NODEINDEPENDENT attribute setting.

Recovery. Perform one of these:

- SET PATHWAY NODEINDEPENDENT to OFF and cold start the PATHWAY object.
- Delete the node name value specified by the `CMDVOL` command.
- Change the node name specified by the `CMDVOL` command to *.

2079

```
*2079* ILLEGAL CPU WEIGHT
```

Cause. The CPU weight is not in the range of 1 through 100. Or, summation of all the weights is not equal to 100.

Effect. The command fails.

Recovery. Redefine the CPU weight with a correct value.

2080

```
*2080* ILLEGAL REBALANCE MODE
```

Cause. Pathway attribute SPREBALANCEMODE is not MANUAL, AUTO, or DISABLE.

Effect. Pathway attribute SPREBALANCEMODE will not be set.

Recovery. Set SPREBALANCEMODE as MANUAL, AUTO, or DISABLE.

2081

```
*2081* ILLEGAL LINK ALGORITHM VALUE
```

Cause. Pathway attribute GLINKALGORITHM or server attribute LINKALGORITHM is not DEFAULT or REQUESTERCPU.

Effect. Pathway attribute GLINKALGORITHM or server attribute LINKALGORITHM will not be set.

Recovery. Set GLINKALGORITHM or LINKALGORITHM as DEFAULT or REQUESTERCPU.

Link Manager Messages (Numbers 3000-3999)

This section describes the messages logged by the PATHMON process on behalf of link managers—the LINKMON process, 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. Message numbers identify the process that reported the error, as follows:

Message Numbers in This Range…	Are Generated by…
1000 through 1499	PATHMON process
1500 through 1999	PDMCOM process
2000 through 2999	PATHCOM process
3000 through 3999	Link managers (TCPs or LINKMON processes)

Messages 3000 through 3999 are described in this section. Messages 2000 through 2999 are discussed in **PATHCOM Messages (Numbers 2000-2999)** on page 289. Messages 1500 through 1999 are discussed in **PDMCOM Messages (Numbers 1500-1999)** on page 281. Messages 1000 through 1499 are described in **PATHMON Messages (Numbers 1000-1499)** on page 245.

This manual also includes log messages generated by LINKMON processes, described in **LINKMON Log Messages** on page 306.

Note that the majority of messages in the 3000 to 3999 range are logged on behalf of Terminal Control Processes (TCPs), the link managers that run under the Pathway/iTS product. This manual lists the messages in the 3000-3999 range that can also be logged on behalf of the LINKMON process, If your environment includes Pathway/iTS and you get an error message not covered in this manual, see the *Pathway/iTS System Management Manual*.

LINKMON Messages

These messages can have the LINKMON process as their subject.

3100

```
*3100* link manager-name, ERROR DURING SERVER OPEN (errnum)
```

Cause. A file-system error occurred while a server process was being opened.

Effect. The operation fails.

Recovery. For information regarding the specified file-system error, see the *Guardian Procedure Errors and Messages Manual*.

3116

```
*3116* link manager-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 link manager unlinks from the server class.

Recovery. For information regarding the specified file-system error, see to the *Guardian Procedure Errors and Messages Manual*.

3117

```
*3117* link manager-name, SERVER CLASS UNDEFINED
```

Cause. The server class specified in a Pathsend statement is not defined for the PATHMON environment.

Effect. The operation fails.

Recovery. Isolate and correct the problem within the requestor program. See the *TS/MP 2.5 Pathsend and Server Programming Manual*.

3118

```
*3118* link manager-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 Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages 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.

3200

```
*3200* link manager-name, INVALID FORMAT MESSAGE RECEIVED BY
TCP
```

Cause. A LINKMON process or terminal control process (TCP) received an invalid request from the PATHMON process. This error usually occurs when incompatible versions of the PATHMON process and the LINKMON process or TCP are running at the same time.

Effect. The request cannot be processed.

Recovery. Use versions of the PATHMON process and the LINKMON process (or TCP) that are compatible with each other. If the problem persists, contact your Hewlett Packard Enterprise representative and provide all relevant information as follows:

- Descriptions of the problem and accompanying symptoms
- Details from the messages 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.

3201

```
*3201* link manager-name, SYNCID VIOLATION IN MESSAGE
RECEIVED BY TCP
```

Cause. A LINKMON process (or TCP) received an invalid message synchronous ID (SYNCID). This is an internal error.

Effect. The request cannot be processed.

Recovery. Contact your Global Customer Support Center (GCSC).

3226

```
*3226* link manager-name, REQUESTED FUNCTION NOT SUPPORTED IN
THIS RELEASE
```

Cause. The PATHMON process requested a function that is not supported in this version of the LINKMON process.

Effect. The operation fails.

Recovery. Use a newer version of the LINKMON process.

3233

```
*3233* link manager-name, SERVER PROCESS UNKNOWN
```

Cause. A LINKMON process received a message from the PATHMON process containing a reference to a server process that is unknown to the link manager. This is an internal error.

Effect. The request cannot be processed.

Recovery. Contact Hewlett Packard Enterprise Support.

LINKMON Log Messages

This section describes, in alphabetic order, the log messages that a LINKMON process can write to the system log, $0, during LINKMON initialization.

General Information

The LINKMON process in a processor does not begin initialization until it receives a SERVERCLASS_SEND_ call from a Pathsend process executing in its processor. If an error occurs during LINKMON initialization, the LINKMON process:

- Returns an error to the application (Pathsend error 947, file-system error).
- Writes a message to $0 indicating the reason for failure.
- Waits ten seconds before processing any additional SERVERCLASS_SEND_ calls, so that subsequent sends to the uninitialized LINKMON process do not flood $0.

Each subsequent SERVERCLASS_SEND_ call in this processor causes the LINKMON process to reattempt initialization; after the error condition is corrected, the LINKMON process can complete initialization.

The LINKMON log messages are type 512 event messages that you can display from $0 with an EMS printing distributor or ViewPoint. For more information, see the *EMS Manual* or the *ViewPoint Manual*. Although type 512 messages are sent to the EMS collector $0, they are not included in the *Operator Messages Manual*; that manual contains only numbered messages.

Some LINKMON messages have file-system errors associated with them. For a description of file-system errors, see the *Guardian Procedure Errors and Messages Manual*.

All LINKMON log messages have the following format:

```
LINKMON ( LINKMON-process-name | CPU,PIN ) - message-text
```

where **message-text** is one of the messages described in this section.

LINKMON Log Messages

These messages are generated by LINKMON processes:

```
Failed to get length bytes of extended memory
```

Cause. The LINKMON process was unable to obtain sufficient memory to continue running.

Effect. The LINKMON process is unable to process SERVERCLASS_SEND_ calls. This message is always followed by an EMS message that describes why the LINKMON process could not obtain enough memory.

Recovery. Recovery depends on the error that occurs. For example, if the next message is “SEGMENT_ALLOCATE_ error 1 (error detail 43) on swap file $SYSTEM.ZLINKMON.ZZLM04,” you must make additional disk space available before the LINKMON process can initialize itself. Each subsequent SERVERCLASS_SEND_ call in this processor causes the LINKMON process to reattempt initialization; after the condition is corrected, the LINKMON process can complete initialization.

```
Initialized and running
```

Cause. The LINKMON process is operational.

Effect. The LINKMON process is available for processing SERVERCLASS_SEND_ calls from requestors in the processor.

Recovery. Informational message only; no corrective action is needed.

```
LINKMON cannot be run
```

Cause. You attempted to start a LINKMON process with a `RUN` command.

Effect. The LINKMON process terminates processing.

Recovery. There is no way to recover from this error condition; you cannot start a LINKMON process with a `RUN` command.

```
SEGMENT_ALLOCATE_ error error-number [ (error detail error-detail) ] on swap file swap-file-
name
```

Cause. An error occurred when the LINKMON process attempted to create an extended memory segment. If SEGMENT_ALLOCATE_ returned error 1, 2, 3, or 14, the message includes error detail information.

Effect. The LINKMON process is unable to process SERVERCLASS_SEND_ calls.

Recovery. Recovery depends on the file error that occurs. For example, error 1 with error detail FENODISCSPACE (43) indicates that the specified disk does not contain enough free space for the swap file. You must make additional disk space available before the LINKMON process can initialize itself. Each subsequent SERVERCLASS_SEND_ call in this processor causes the LINKMON process to reattempt initialization; after the condition is corrected, the LINKMON process can complete initialization.

```
Terminating
```

Cause. The LINKMON process has terminated.

Effect. The LINKMON process is no longer able to support SERVERCLASS_SEND_ calls in its processor.

Recovery. This message is not expected during normal operation. If you receive this message, contact your Hewlett Packard Enterprise representative.

```
Unable to purge existing swap file swap-file-name
(file purge error error-number)
```

Cause. A file-system error occurred when the LINKMON process attempted to purge an old extended memory swap file.

Effect. The LINKMON process is unable to process SERVERCLASS_SEND_ calls.

Recovery. This message is not expected during normal operation. Recovery depends on the file error that occurs. Each subsequent SERVERCLASS_SEND_ call in this processor causes the LINKMON process to reattempt initialization; after the condition is corrected, the LINKMON process can complete initialization.

```
Unable to secure swap file file-name ( reason )
```

Cause. A file-system error occurred when the LINKMON process attempted to secure its extended memory swap file.

Effect. The LINKMON process is unable to process SERVERCLASS_SEND_ calls.

Recovery. This error is not expected during normal operations. Recovery depends on the file error that occurs. For example, if you receive a file in use error (12), ensure that the file is available before issuing another SERVERCLASS_SEND_ call in this processor. Each subsequent SERVERCLASS_SEND_ call in this processor causes the LINKMON process to reattempt initialization; after the condition is corrected, the LINKMON process can complete initialization.

ACS Subsystem Messages

Introduction

NOTE: The messages described in the section are a subset of the messages in the messagenumber range 1000 through 1071 that are reported by ACS subsystem processes. The messages reported by TCPs are described in the *Pathway/iTS Management Programming Manual*.

All messages returned by the PATHMON environment are logged by the PATHMON process. However, some messages in some number ranges represent events and errors reported to the PATHMON process by other processes. The message number identifies the process that reported the event, as:

ACS Subsystem Identifiers

ACS subsystem number for NonStop OS SPI subsystems	252
ACS subsystem name in the NonStop OS	APPCLSTR
ACS subsystem name abbreviation	ACL
External ACS subsystem identifier for release TS/MP 2.6 or later. This identifier occurs in event message text, which is typically produced by EMS event printing distributors for EMS formatted event messages logged by ACS subsystem processes.	TANDEM.APPCLSTR.H09

Event-Message SPI Format

Each event message in this chapter includes:

- The number and name of the event message
- A description of the event-message cause
- A syntax box listing the tokens associated with the message
- Token descriptions
- Effect and recovery information

The syntax box contains the text version of the event message, which is available through the EMSTEXT procedure. When you use the EMSTEXT procedure to get the text version of an event message, you request either the display format or the console compatible format, as described in the *EMS Manual*. For the ACS subsystem event messages, both the versions are identical.

Listed Tokens

Conditional tokens and unconditional tokens that are specific to an event message are included in the message descriptions.

Tokens specific to the ACS subsystem begin with the unique subsystem identifier ZACL- (the identifier ZACS- is used by a different subsystem).

Unlisted Tokens

When a token is common to all event messages, it is not included in the message descriptions in this chapter. For a description of all tokens not listed in this chapter, see the *TS/MP Management Programming Manual*.

The following unconditional tokens, which can have different values for each event message, occur in each event message generated by the ACS subsystem and are not individually listed in the message descriptions:

ZEMS-TKN-EMPHASIS **token-type ZSPI-TYP-BOOLEAN**

Contains the value ZSPI-VAL-FALSE if the value of ZACL-TKN-EVENT-LEVEL is less than ZACL-VAL-EVENT-LEVEL-WARNING.

Contains the value ZSPI-VAL-TRUE if the value of ZACL-TKN-EVENT-LEVEL is greater than or equal to ZACL-VAL-EVENT-LEVEL-WARNING.

ZEMS-TKN-EVENTNUMBER **token-type ZSPI-TYP-INT**

Contains the ACS subsystem event message number.

ZEMS-TKN-TEXT **token-type ZSPI-TYP-STRING**

Table Continued

Contains the message text for each event message. Placeholders for conditional tokens that are omitted from an event message are replaced by empty strings. Null values of tokens included in the event message are converted to strings similar to non-null values.

The value of this token is used only when no EMS message template for the event is installed on the system in which the event message is generated, or when a filesystem error prevents the EMS event message formatting facilities from retrieving the message template.

ZSPI-TKN-MANAGER **token-type ZSPI-TYP-FNAME32**

Contains the name of the subsystem manager process responsible for the subject of the event message. If the subject token of an event is ZACL-TKN-EVENT-SERVER-NAME, the value of ZSPI-TKN-MANAGER is the name of the PATHMON process that manages the named server class. Otherwise, the value of ZSPI-TKN-MANAGER is the name of the ACS domain coordinator process, $ZACS, on the system where the event originated.

ACS Event Severity Levels

Event severity levels indicate the impact (on ACS subsystem resources and processes) of an event occurrence detected by an ACS subsystem process. Every event occurrence logged by the ACS subsystem indicates the severity of that occurrence. Most events defined by the ACS subsystem can occur with any one of several severity levels depending upon the context in which the event occurs and on how well the ACS subsystem can recover from the occurrence.

Definition ZACL-DDL-EVENT-LEVEL Type Enum Tacl Enum

Begin			
89	ZACL-VAL-EVENT-LEVEL-INFO	Value 0	As "INFO".
89	ZACL-VAL-EVENT-LEVEL-STATUS	Value 1	As "STATUS".
89	ZACL-VAL-EVENT-LEVEL-WARNING	Value 2	As "WARNING".
89	ZACL-VAL-EVENT-LEVEL-ERROR	Value 3	As "ERROR".
89	ZACL-VAL-EVENT-LEVEL-FATAL	Value 4	As "FATAL".
End			

Token -Type	
ZACL-TYP-EVENT-LEVEL	Value is ZSPI-TDT-ENUMDef is ZACL-DDL-EVENT-LEVEL.

`ZACL-VAL-EVENT-LEVEL-INFO`

The ACS subsystem does not generate info events. This event level is reserved for future use.

`ZACL-VAL-EVENT-LEVEL-STATUS`

Table Continued

Indicates that the ACS subsystem made an orderly change in the state or status of an object, resource, or process under control or it detected an orderly change in the state of an external resource or process. No recovery actions are required for status events.

ZACL-VAL-EVENT-LEVEL-WARNING

Indicates that the ACS subsystem detected a possible problem in a Pathway application or it detected a condition that might cause errors in ACS subsystem processes in the future if not corrected. Recovery or preventive actions are advised for warning events.

ZACL-VAL-EVENT-LEVEL-ERROR

Indicates that resources or processes required by the ACS subsystem have failed or an ACS subsystem process encountered a condition that prevented it from completing an operation. The operation, if any, is aborted but the process that generated the event message continues to accept requests and perform other operations. Typically, error events require correcting the error condition and then retrying the operation.

ZACL-VAL-EVENT-LEVEL-FATAL

Indicates that an ACS subsystem process encountered a condition that prevented it from performing any more operations. The process terminates immediately, which aborts any operations that it had in progress. If the condition was an internal error in the process logic or data, it produces a SaveAbend file. Typically, the process termination disables the CPU in which it was running. Recovery actions, including restarting the ACS subsystem, are required for fatal events.

EMS Message Range 1000 through 1009

1001: ZACL-EVT-BACKUP-LAUNCHED

The backup domain coordinator process was successfully created in the indicated CPU.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL      token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.
```

Event Message Text

```
1001 - Backup process created in CPU cpu-number
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL

has the value ZACL-VAL-EVENT-LEVEL-STATUS.

ZACL-TKN-EVENT-CPU-NUMBER

contains the number of the CPU on the local system in which the backup domain coordinator process was started. This token is the subject of the event message.

Effect. The backup domain coordinator process runs in the indicated CPU.

Recovery. This is an informational message only; no corrective action is needed.

1002: ZACL-EVT-BACKUP-LAUNCH-ERROR

The domain coordinator process cannot create a backup process in the specified CPU on the local system because the PROCESS_LAUNCH_() operation failed.l

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL        token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER   token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-CODE   token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT
```

Event Message Text

```
1002 - Backup proc.ess creation in CPU cpu-number failed,
error errnum, detail error-detail
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZACL-TKN-EVENT-CPU-NUMBER`

contains the number of the CPU on the local system in which the backup domain coordinator process could not be started. This token is the subject of the event message.

`ZACL-TKN-EVENT-ERROR-CODE`

contains the process-creation error code returned by the PROCESS_LAUNCH_() procedure.

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains the process-creation error detail code returned by the PROCESS_LAUNCH_() operation. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message.

Effect. The domain coordinator process runs as a single process.

Recovery. Correct the error condition. If the problem persists, contact your Hewlett Packard Enterprise support analyst.

1003: ZACL-EVT-BACKUP-ABENDED

The backup domain coordinator process terminated unexpectedly.]

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER       token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-CPU-NUMBER       token-type ZSPI-TYP-INT.
```

Conditional Tokens

Table Continued

```
ZACL-TKN-EVENT-TERM-STOPPER token-type ZSPI-TYP-PHANDLE.
ZACL-TKN-EVENT-TERM-SSID               token-type ZSPI-TYP-SSID.
ZACL-TKN-EVENT-TERM-TEXT               token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1003 - Backup process in CPU cpu-number terminated
abnormally, comp-code error-code, [ stopper stopper-id, ] [
term-ssid term-ssid, ] term-info term-info, [ term-text termtext
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZACL-TKN-EVENT-CPU-NUMBER`

contains the number of the CPU on the local system in which the backup domain coordinator process was terminated. This token is the subject of the event message.

`ZACL-TKN-EVENT-ERROR-CODE`

contains the completion code associated with the backup process termination. A completion code of -4 indicates that only the backup process, not both members of the process pair, terminated.

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains the termination information code returned from the process-termination operation. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message. If the backup process was stopped externally (indicated by completion code 6), the value of this token is the creator user identifier of the external process that stopped the backup process.

`ZACL-TKN-EVENT-TERM-STOPPER`

contains the process handle of the process that initiated the termination. This token is included only if the process was stopped externally.

`ZACL-TKN-EVENT-TERM-SSID`

contains the SPI subsystem identifier associated with the process-termination operation. This token is omitted if no subsystem identifier was associated with the termination.

`ZACL-TKN-EVENT-TERM-TEXT`

contains the termination text associated with the process-termination operation. This token is omitted if no termination text was associated with the termination.

Effect. The primary domain coordinator process creates a new backup process.

Recovery. If the problem persists, contact your Hewlett Packard Enterprise support analyst.

1004: ZACL-EVT-CP-LAUNCH-ERROR

Either the domain coordinator process cannot create an ACS subsystem process in the specified CPU on the local system because the PROCESS_LAUNCH_() operation failed or the ACS subsystem process was created but the PROCESS_LAUNCH_() operation returned a warning.]

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER       token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-PROCESS-NAME     token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE       token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL     token-type ZSPI-TYP-INT.
```

Conditional Token

```
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1004 - Core process creation in CPU cpu-number failed, name
process-name, error errnum, detail error-detail, [ file filename
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING, ZACL-VAL-EVENTLEVEL- ERROR, or ZACL-VAL-EVENT-LEVEL-FATAL.

`ZACL-TKN-EVENT-CPU-NUMBER`

contains the number of the CPU on the local system in which the process could not be started. This token is the subject of the event message.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the process that was not created.

`ZACL-TKN-EVENT-ERROR-CODE`

contains the process-creation error code returned by the PROCESS_LAUNCH_() procedure.

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains additional information about the process-creation error returned by the PROCESS_LAUNCH_() operation. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message.

`ZACL-TKN-EVENT-FILE-NAME`

contains the Guardian name of a program file, library file, swap file, home terminal, or process name associated with the major error code returned by the PROCESS_LAUNCH_() operation. This token is omitted if no file is associated with the major error code.

Effect. No ACS subsystem processes are started in the specified CPU. If the PROCESS_LAUNCH_() operation returned an error indicating a programming error in the domain coordinator process, the event is fatal; the primary domain coordinator process terminates abnormally and produces a SaveAbend file.

Recovery. Correct the condition that caused the process creation failure then use the `CONTROL ACS` command to restart the ACS subsystem processes in the CPU. If the event is fatal or the problem persists, contact your Hewlett Packard Enterprise support analyst.

1005: ZACL-EVT-CP-ABENDED

An ACS subsystem process terminated unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT.
```

Conditional Tokens

```
ZACL-TKN-EVENT-TERM-STOPPER token-type ZSPI-TYP-SSID.
ZACL-TKN-EVENT-TERM-SSID token-type ZSPI-TYP-PHANDLE.
ZACL-TKN-EVENT-TERM-TEXT token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1005 - Core process in CPU cpu-number terminated abnormally,
name process-name, comp-code comp-code, [ stopper stopper-id,
] [ term-ssid term-ssid, ] term-info term-info, [ term-text
term-text
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-ERROR.

`ZACL-TKN-EVENT-CPU-NUMBER`

contains the number of the CPU on the local system in which the process terminated. This token is the subject of the event message.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the terminated process.

`ZACL-TKN-EVENT-ERROR-CODE`

contains the completion code associated with the process-termination operation.

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains the termination information code associated with the process termination. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message. If the process was stopped externally (indicated by completion code 6), the value of this token is the creator user identifier of the external process that initiated the termination.

ZACL-TKN-EVENT-TERM-STOPPER

Table Continued

contains the process handle of the process that caused the termination. This token is included only if the process was stopped externally.

```
ZACL-TKN-EVENT-TERM-SSID
```

contains the SPI subsystem identifier associated with the process-termination. This token is omitted if no subsystem identifier was associated with the termination.

```
ZACL-TKN-EVENT-TERM-TEXT
```

contains the termination text associated with the process-termination. This token is omitted if no termination text was associated with the termination.

Effect. The ACS subsystem process is not available in the indicated CPU; other ACS subsystem processes continue to run on the CPU.

Recovery. Correct the condition that caused the file error, and retry the operation.

1006: ZACL-EVT-FILE-ERROR

An ACS subsystem process invoked a Guardian file-system procedure on an open or named file and the operation failed unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
```

Conditional Token

```
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1006 - Guardian file operation procedure() failed, [ file
file-name, ] error errnum
```

Token Descriptions

```
ZACL-TKN-EVENT-LEVEL
```

has one of the values ZACL-VAL-EVENT-LEVEL-WARNING, ZACL-VAL-EVENTLEVEL- ERROR, or ZACL-VAL-EVENT-LEVEL-FATAL.

```
ZACL-TKN-EVENT-PROCEDURE
```

the name of the Guardian file-system procedure that reported the failure. This token is the subject of the event message.

```
ZACL-TKN-EVENT-PROCESS-NAME
```

contains the name of the failed process.

Table Continued

```
ZACL-TKN-EVENT-ERROR-CODE
```

contains the file-system error code returned by the failed operation.

```
ZACL-TKN-EVENT-FILE-NAME
```

contains the name of the Guardian file, device, or process accessed by the failed operation. This token is omitted for Guardian procedures that do not operate on files but that, nevertheless, report failures using file-system error codes.

Effect. The operation associated with the file-system operation is aborted.

Recovery. Correct the condition that caused the file error, and retry the operation.

1007: ZACL-EVT-OSS-ERROR

An ACS subsystem process invoked an OSS procedure and the operation failed unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-OSS-ERRNO token-type ZSPI-TYP-INT2.
```

Conditional Token

```
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1007 - OSS operation procedure() failed, pathname pathname,
error oss-errno
```

Token Descriptions

```
ZACL-TKN-EVENT-LEVEL
```

has one of the values ZACL-VAL-EVENT-LEVEL-WARNING, ZACL-VAL-EVENTLEVEL- ERROR, or ZACL-VAL-EVENT-LEVEL-FATAL.

```
ZACL-TKN-EVENT-PROCEDURE
```

contains the name of the OSS procedure that reported the failure. This token is the subject of the event message.

```
ZACL-TKN-EVENT-OSS-ERRNO token-type ZSPI-TYP-INT2
```

contains the file OSS error number returned by the failed operation.

```
ZACL-TKN-EVENT-FILE-NAME
```

contains the path name of the OSS file or directory that was accessed by the failed operation. This token is omitted for OSS procedures that do not operate on files or directories.

Effect. The operation associated with the OSS operation is aborted.

Recovery. Correct the condition that caused the OSS error, and retry the operation.

1008: ZACL-EVT-PROCINFO-ERROR

An ACS subsystem process invoked a Guardian process information procedure and the operation failed unexpectedly. I

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1008 - Process operation procedure() failed, process processname,
error errnum, detail error-detai
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-ERROR.

`ZACL-TKN-EVENT-PROCEDURE`

contains the name of the Guardian process management procedure that reported the failure. This token is the subject of the event message.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the ACS subsystem process associated with the failure.

`ZACL-TKN-EVENT-ERROR-CODE`

contains the name of the ACS subsystem process associated with the failure.

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains additional information about the error returned by the failed operation. If the value of this token is zero in an event message, the token is null and must be regarded as omitted from the message.

Effect. The operation associated with the process-management operation is aborted.

Recovery. Correct the condition that caused the error, and retry the operation.

1009: ZACL-EVT-PATHSEND-ERROR

An ACS subsystem process invoked an internal Pathsend or ACS subsystem link manager procedure and the operation failed unexpectedly. This event message is generated only for ACS subsystem processes; it is not generated for application Pathsend requestors that encounter failures when they invoke Pathsend API procedures.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT.
```

Conditional Token

```
ZACL-TKN-EVENT-SERVER-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1009 - Pathsend operation procedure() failed, server servername,
error errnum, detail error-detai
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-ERROR.

`ZACL-TKN-EVENT-PROCEDURE`

contains the name of the Pathsend or link-management procedure that reported the failure. This token is the subject of the event message.

`ZACL-TKN-EVENT-ERROR-CODE`

contains the Pathsend error code returned by the failed operation.

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains the file-system error code returned by the failed operation. If the value of this token is zero in an event message, the token is null and is omitted from the message.

`ZACL-TKN-EVENT-SERVER-NAME`

contains the PATHMON process name and server class name supplied to the failed operation, separated by a period. This token is omitted for link manager procedures that do not operate on server classes.

Effect. The ACS subsystem operation associated with the failed operation is aborted.

Recovery. Correct the condition that caused the error, and retry the operation.

EMS Messages 1010 through 1019

1010: ZACL-EVT-SEGMENT-ERROR

An ACS subsystem process invoked a Guardian memory-segment management procedure and the operation failed unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-SEGMENT-ID token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1010 - Memory segment operation procedure() failed, segment
segment-id, error errnum, detail error-detai
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL. ZACL-TKN-EVENT-PROCEDURE contains the name of the Guardian memory-segment management procedure that reported the failure. This token is the subject of the event message. ZACL-TKN-EVENT-SEGMENT-ID contains the segment identifier of the memory segment being accessed by the failed operation. ZACL-TKN-EVENT-ERROR-CODE contains the memory management error code returned by the failed operation. ZACL-TKN-EVENT-ERROR-DETAIL contains the memory management error detail code returned by the failed operation. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message.

Effect. The process that invoked the failed operation terminates abnormally and produces a SaveAbend file.

Recovery. If the failure was due to a file-system error or system-resource limit, correct the condition and use the `CONTROL ACS` command to restart the ACS subsystem processes in the CPU where the failure occurred.

If this is an internal error; contact your Hewlett Packard Enterprise analyst.

1011: ZACL-EVT-SEGMENT-CORRUPT

An ACS subsystem process detected corruption in the ACS subsystem shared memory segment in the CPU.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL
ZACL-TKN-EVENT-SEGMENT-ID token-type ZSPI-TYP-INT.
```

Table Continued

Event-Message Text

```
1011 - Corruption detected in shared memory segment segmentid
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL. ZACL-TKN-EVENT-SEGMENT-ID contains the segment identifier of the memory segment in which the corruption was detected. This token is the subject of the event message.

Effect. The process that detected the shared-memory segment corruption terminates abnormally and produces a SaveAbend file.

Recovery. Use the `CONTROL ACS` command to restart the ACS subsystem processes in the CPU where the corruption was detected. If the problem persists, contact your Hewlett Packard Enterprise support analyst.

1012: ZACL-EVT-BINSEM-ERROR

An ACS subsystem process invoked a Guardian semaphore-system procedure and the operation failed unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL        token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCEDURE    token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-SEGMENT-ID   token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-CODE   token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1012 - Memory semaphore operation procedure() failed, segment
segment-id, error errnum
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL. ZACL-TKN-EVENT-PROCEDURE contains the name of the Guardian binary semaphore procedure that reported the failure. This token is the subject of the event message. ZACL-TKN-EVENT-SEGMENT-ID

Table Continued

contains the identifier of the memory segment protected by the semaphore that was being accessed by the failed operation.

ZACL-TKN-EVENT-ERROR-CODE

contains the Guardian or OSS error code returned by the failed operation.

Effect. The process that invoked the failed operation terminates abnormally and produces a SaveAbend file.

Recovery. This is an internal error. Contact your Hewlett Packard Enterprise analyst.

1014: ZACL-EVT-BINSEM-ABANDONED

A process terminated while holding the lock on one of the ACS subsystem shared memory semaphores.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-SEGMENT-ID token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1014 - Process failed holding memory segment segment-id
semaphore
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL

has the value of ZACL-VAL-EVENT-LEVEL-FATAL.

ZACL-TKN-EVENT-SEGMENT-ID

contains the identifier of the memory segment protected by the semaphore that was being accessed by the failed operation. This token is the subject of the event message.

Effect. The process that detected the abandoned semaphore terminates immediately. The shared memory might be corrupt.

Recovery. Use the `CONTROL ACS` command to restart the ACS subsystem processes in the CPU where the semaphore was abandoned.

1015: ZACL-EVT-MEMPOOL-ERROR

An ACS subsystem process invoked a Guardian memory-pool management procedure and the operation failed unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-SEGMENT-ID token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
```

Table Continued

Event-Message Text

```
1015 - Memory pool operation procedure() failed, segment
segment-id, error errnum
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL. ZACL-TKN-EVENT-PROCEDURE contains the name of the Guardian memory-pool management procedure that reported the failure. This token is the subject of the event message. ZACL-TKN-EVENT-SEGMENT-ID contains the identifier of the memory segment containing the pool that was being accessed by the failed operation. ZACL-TKN-EVENT-ERROR-CODE contains the error code returned by the failed operation.

Effect. The process that invoked the failed operation terminates abnormally and produces a SaveAbend file.

Recovery. This is an internal error; contact your Hewlett Packard Enterprise analyst.

Refer to the *Operator Messages Manual*, Appendix B, File-System Errors, for a definition of the specified error. For more detailed information, including recovery actions, refer to the *Guardian Procedure Errors and Messages Manual*.

1016 ZACL-EVT-MEMPOOL-NOSPACE

An ACS subsystem process could not allocate the space it needed in the ACS shared memory segment in the CPU. Depending upon why the process needed the memory allocation, the operation it was performing might have failed or the ACS subsystem processes in the CPU might need to be restarted.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-SEGMENT-ID token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1016 - Insufficient free space in shared memory segment-id
pool
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL.

Table Continued

ZACL-TKN-EVENT-SEGMENT-ID contains the identifier of the memory segment containing the pool from which space was to be allocated. This token is the subject of the event message.

Effect. The process that failed to allocate space from the memory segment terminates immediately.

Recovery. Use the `CONTROL ACS` command to restart the ACS subsystem processes in the CPU where the problem occurred.

1017: ZACL-EVT-UNAUTH-REQUESTOR

An unauthorized process attempted to open an ACS subsystem process.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL      token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-ID token-type ZSPI-TYP-PHANDLE.
ZACL-TKN-EVENT-ACCESS-ID  token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-CREATOR-ID token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-FILE-NAME  token-type ZSPI-TYP-STRING.
```

Conditional Token

```
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1017 - Unauthorized OPEN attempted by process process-id,
access-user access-id, creator-user creator-id, process-name
process-name, open-file file-name
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-PROCESS-ID contains the process handle of the process that sent the unauthorized OPEN request message. This token is the subject of the event message. If the process identified by this token still exists when the event message text is generated, the EMS event message formatting facilities convert process handle tokens to named or unnamed process descriptor strings. If the process no longer exists, EMS generates a question mark instead of the process descriptor string. ZACL-TKN-EVENT-ACCESS-ID contains the process-access user identifier of the unauthorized requestor process. ZACL-TKN-EVENT-CREATOR-ID contains the process-creator user identifier of the unauthorized requestor process.

Table Continued

ZACL-TKN-EVENT-FILE-NAME

contains the file name of the ACS subsystem process that the unauthorized process supplied to the OPEN or FILE_OPEN_ operation.

ZACL-TKN-EVENT-PROCESS-NAME

contains the named process descriptor of the process that sent the unauthorized OPEN request. This token is omitted if the unauthorized process is unnamed.

Effect. The attempt fails with an FESECVIOL (48) file-system error.

Recovery. None. This event indicates a possible attempt to breach the security of the ACS subsystem.

1018: ZACL-EVT-INVALID-RQST-MSG

An ACS subsystem process received an invalid request message.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-ID token-type ZSPI-TYP-PHANDLE.
```

Event-Message Text

```
1018 - Invalid request message received from process process-id
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

ZACL-TKN-EVENT-PROCESS-ID

contains the process handle of the requestor process that sent the invalid request message. This token is the subject of the event message. If the process identified by this token still exists when the event message text is generated, the EMS event message formatting facilities convert process handle tokens to named or unnamed process descriptor strings. If the process no longer exists, EMS generates a question mark instead of the process descriptor string.

Effect. The invalid request message is ignored. A reply containing an FEINVALOP (2) error is returned to the requestor process.

Recovery. Correct the requestor program.

1019: ZACL-EVT-INCOMP-RQST-VER

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL
ZACL-TKN-EVENT-PROCESS-ID token-type ZSPI-TYP-PHANDLE
```

Event-Message Text

```
1019 - Incompatible request version received from process
process-id
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-PROCESS-ID contains the process handle of the requestor process that sent the incompatible request message. This token is the subject of the event message. If the process identified by this token still exists when the event message text is generated, the EMS event message formatting facilities convert process handle tokens to named or unnamed process descriptor strings. If the process no longer exists, EMS generates a question mark instead of the process descriptor string.

Cause. An ACS subsystem process received an incompatible version of a request message.

Effect. The request message is ignored. A reply containing an FEINVALOP (2) error is returned to the requestor process.

Recovery. This event indicates that incompatible versions of ACS components are installed on different systems in the same domain.

EMS Messages 1020 through 1029

1020: ZACL-EVT-INVALID-RPLY-MSG

An ACS subsystem process received an invalid reply message or a reply message that has an incompatible version.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-ID token-type ZSPI-TYP-PHANDLE.
```

Event-Message Text

```
1020 - Invalid reply message received from process process-id
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has one of the values ZACL-VAL-EVENT-LEVEL-ERROR or ZACL-VAL-EVENTLEVEL- FATAL. ZACL-TKN-EVENT-PROCESS-ID contains the process handle of the server process that sent the invalid reply message. This token is the subject of the event message. If the process identified by this token still exists when the event message text is generated, the EMS event message formatting facilities convert process handle tokens to named or unnamed process descriptor strings. If the process no longer exists, EMS generates a question mark instead of the process descriptor string.

Effect. The reply message is discarded. The operation associated with the request is aborted.

Recovery. Contact your Hewlett Packard Enterprise support analyst.

1021: ZACL-EVT-PROTOCOL-ERROR

An ACS subsystem process detected a protocol (operation order) error by another system process.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-ID token-type ZSPI-TYP-PHANDLE.
```

Event-Message Text

```
1021 - Protocol error in communication with process processid
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-ERROR.

`ZACL-TKN-EVENT-PROCESS-ID`

contains the process handle of the process that committed the protocol error. This token is the subject of the event message.

Effect. The operation in which the protocol error occurred is aborted.

Recovery. Contact your Hewlett Packard Enterprise support analyst.

1022: ZACL-EVT-PATHMON-IO-TIMEOUT

An I/O operation from an ACS subsystem process to a PATHMON process timed out. ACS subsystem processes use a timeout value of 5 minutes for an I/O to a PATHMON process.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1022 - PATHMON process-name I/O operation timed out
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the PATHMON process. This token is the subject of the event message.

Effect. If the I/O operation was initiated in response to a send operation, the send operation fails with Pathsend error FEScPathmonConnect and file-system error FETIMEDOUT.

The ACS subsystem attempts to close communication to this PATHMON process. New send requests fail with Pathsend error FEScPathmonShutdown and file-system error FEOK.

When all links to the PATHMON process are closed from the requestor side, the ACS subsystem closes communication to this PATHMON process then opens a new communication channel for future send requests.

Recovery. Close all the links from the requestor side to this PATHMON process. After closing all the links, retry send operations. If the problem persists, try to find out why the PATHMON process continues to time out.

1023: ZACL-EVT-TSMP-PROTOCOL-VIOL

The PATHMON process communication with the ACS subsystem does not follow the TSMP protocol.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1023 - TSMP protocol violation by PATHMON process-name
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has either the value ZACL-VAL-EVENT-LEVEL-WARNING or ZACL-VAL-EVENTLEVEL- ERROR.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the PATHMON process. This token is the subject of the event message.

Effect. The send operation fails with Pathsend error FEScPathmonMessage and file-system error FEOK.

Recovery. Retry the send operation. If the problem persists, check the PATHMON process. You must shut down and restart your application.

1024: ZACL-EVT-PATHMON-IO-ERROR

An I/O operation from the ACS subsystem to a PATHMON process failed unexpectedly.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1024 - PATHMON process-name I/O operation failed, error
errnum
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-PROCESS-NAME contains the name of the PATHMON process. This token is the subject of the event message. ZACL-TKN-EVENT-ERROR-CODE contains the file-system error code returned by the failed operation.

Effect. If the I/O operation was made in response to a send operation, the send operation fails with Pathsend error FEScPathmonConnect. The file-system error code provides detailed information on the error.

The ACS subsystem attempts to close communication to this PATHMON process. New send requests fail with Pathsend error FEScPathmonShutdown and file-system error FEOK.

When all links to the PATHMON process are closed from the requestor side, the ACS subsystem closes communication to this PATHMON process then opens a new communication channel for future send requests.

Recovery. Close all the links to this PATHMON process from the requestor side.

1025: ZACL-EVT-PATHMON-OPEN-ERROR

An attempt to open a PATHMON process failed.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1025 - PATHMON process-name OPEN operation failed, error
errnum
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-PROCESS-NAME contains the name of the PATHMON process. This token is the subject of the event message. ZACL-TKN-EVENT-ERROR-CODE contains the file-system error code returned by the failed operation.

Effect. The send operation fails with Pathsend error FEScPathmonConnect.

Recovery. Check the error code for detailed information about the error. Retry the send operation. If the problem persists, contact your Hewlett Packard Enterprise support analyst.

1026: ZACL-EVT-PATHMON-NOOPEN

A PATHMON process could not open an ACS subsystem process.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1026 - No OPEN received from PATHMON process-name
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the PATHMON process. This token is the subject of the event message.

Effect. The send operation fails with Pathsend error FEScPathmonMessage and filesystem error FEOK.

Recovery. Try the send operation again. If the problem persists, check the PATHMON process. You must shut down and restart your application.

1027: ZACL-EVT-PATHMON-EXT-SHUTDOWN

A PATHMON process sent a PATHMONSHUTDOWN request to the ACS subsystem. This condition typically occurs in response to a `TS/MP SHUTDOWN` command.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1027 - PATHMON process-name was externally shut down
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the PATHMON process. This token is the subject of the event message.

Effect. Depending upon the availability of the links with the ACS subsystem processes, some of the send operations might succeed using the existing links, and others might fail with Pathsend error FEScPathmonShutdown and file-system error FEOK.

Recovery. No recovery action is required.

1028: ZACL-EVT-PATHMON-INVALID-MAT

An ACS subsystem process received an invalid message from a PATHMON process.

Unconditional Tokens

`ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL`
`ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.`
 	
Event-Message Text
 	
`1028 - Invalid message received from PATHMON process-name`

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. `ZACL-TKN-EVENT-PROCESS-NAME` contains the name of the PATHMON process. This token is the subject of the event message.

Effect. If the MAT message was sent in response to an operational command from PATHCOM or from SPI requestors, the command fails. PATHMON logs an EMS event.

Recovery. No recovery action is required.

1029: ZACL-EVT-CPU-STARTED

The ACS subsystem processes in the specified CPU were started in response to a `START` command or were restarted in response to a `CONTROL` command.

Unconditional Tokens

`ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.`
`ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.`
 	
Event-Message Text
 	
`1029 - Core processes started in CPU cpu-number`

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`
has the value of ZACL-VAL-EVENT-LEVEL-STATUS. `ZACL-TKN-EVENT-CPU-NUMBER` contains the number of the physical CPU that was started on the system where the event message was generated. This token is the subject of the event message.

Effect. Pathsend operations initiated in the CPU no longer fail with FESCLINKMONCONNECT errors.

Recovery. This is an informational message only; no corrective action is needed.

EMS Messages 1030 through 1039

1030: ZACL-EVT-CPU-STOPPED

One or all of the ACS subsystem processes in a CPU on the local system failed or were stopped in response to a `STOP` command. If this error occurred due to a failure, the physical CPU might have failed.

Unconditional Tokens

`ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.`
`ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.`
 	
Event-Message Text
 	
`1030 - Core processes stopped in CPU cpu-number`

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`
has the value of ZACL-VAL-EVENT-LEVEL-ERROR. `ZACL-TKN-EVENT-CPU-NUMBER` contains the number of the physical CPU in which the ACS subsystem processes were stopped on the system where the event message was generated. This token is the subject of the event message.

Effect. Pathsend operations initiated in the CPU fail with FESCLINKMONCONNECT errors until the core processes are restarted.

Recovery. Use the `CONTROL ACS` command to restart the ACS subsystem processes in the CPU.

1031: ZACL-EVT-LOG-FILE-ERROR

An ACS subsystem process encountered an error while writing an event message to the configured LOG1 or LOG2 event log file.

Unconditional Tokens

`ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.`
`ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.`
`ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.`
`ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.`
 	
Event-Message Text
 	
`1031 - Event log file operation procedure() failed, file file-name, error errnum`

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`
has the value of ZACL-VAL-EVENT-LEVEL-ERROR.

Table Continued

ZACL-TKN-EVENT-PROCEDURE contains the name of the Guardian file-system procedure that reported the failure. ZACL-TKN-EVENT-ERROR-CODE contains the file-system error code returned by the failed operation. ZACL-TKN-EVENT-FILE-NAME contains the name of the event log file that was accessed by the failed operation.

Effect. The configured event log is closed. Event messages continue to be logged to the other configured event log file, if it is open, using that event log configured format and filter.

If neither configured event log is open, all events generate messages, which are sent to $0 in EMS format.

If an error occurs while writing to $0, $0 is closed and the event message is discarded. The ACS subsystem attempts to reopen $0 when another event occurs.

Recovery. Correct the condition that caused the error, and then reopen the configured log files by stopping and restarting the ACS subsystem.

1032: ZACL-EVT-LOG1-OPENED

The LOG1 event log file was opened successfully.

Unconditional Tokens
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL. ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING. **Event-Message Text** 1032 - LOG1 opened on file file-name

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-STATUS. ZACL-TKN-EVENT-FILE-NAME contains the Guardian name of the event log file. This token is the subject of the event message.

Effect. None.

Recovery. This is an informational message only; no corrective action is needed.

1033: ZACL-EVT-LOG1-CLOSED

The LOG1 event log file was closed either in response to an operator command or because an error occurred while writing to the log file. If this event was due to an error on the log file, then a previously issued event message described the error.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1033 - LOG1 closed on file file-name
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-STATUS.

`ZACL-TKN-EVENT-FILE-NAME`

contains the Guardian name of the event log file. This token is the subject of the event message.

Effect. None.

Recovery. If the event was due to an error on the log file, then check the previously issued event message for a detailed description of the problem. If the event was issued in response to an operator command, then this is an informational message only and no corrective action is needed.

1034: ZACL-EVT-LOG2-OPENED

The LOG2 event log file was opened successfully.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1034 - LOG2 opened on file file-name
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-STATUS.

`ZACL-TKN-EVENT-FILE-NAME`

contains the Guardian name of the event log file. This token is the subject of the event message.

Effect. None.

Recovery. This is an informational message only; no corrective action is needed.

1035: ZACL-EVT-LOG2-CLOSED

The LOG2 event log file was closed either in response to an operator command or because an error occurred while writing to the log file. If this event was due to an error on the log file, a previously issued event message described the error.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1035 - LOG2 closed on file file-name
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-STATUS.

`ZACL-TKN-EVENT-FILE-NAME`

contains the Guardian name of the event log file. This token is the subject of the event message.

Effect. None.

Recovery. If the event was due to an error on the log file, check the previously issued event message for a detailed description of the problem. If the event was issued in response to an operator command, this is an informational message only; no corrective action is needed.

1036: ZACL-EVT-ENV-NAME-UNDEF

The PATHMON process name supplied to a Pathsend operation is not currently running.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZSPI-TKN-EVENT-ENV-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1036 - PATHMON process process-name is not running
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZSPI-TKN-EVENT-ENV-NAME`

contains the PATHMON process name supplied to the Pathsend operation. This token is the subject of the event message.

Effect. The Pathsend operation fails with an FESCPATHMONCONNECT error and a FENOSUCHDEV file-system error detail.

Recovery. Correct the Pathsend requestor program or start the named PATHMON process.

1037: ZACL-EVT-SERVER-NAME-UNDEF

The server class name supplied to a Pathsend operation is not defined in the PATHMON process specified with it.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-SERVER-NAME token-type ZSPI-TYP-STRING.
ZSPI-TKN-MANAGER token-type ZSPI-TYP-FNAME32.
```

Event-Message Text

```
1037 - Server name pathmon-name.serverclass-name is not
defined
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-WARNING.

`ZACL-TKN-EVENT-SERVER-NAME`

contains the server class name supplied to the Pathsend operation. This token is the subject of the event message.

`ZSPI-TKN-MANAGER`

contains the name of the PATHMON process specified in the Pathsend operation.

Effect. The Pathsend operation fails.

Recovery. Correct the Pathsend requestor program.

1038: ZACL-EVT-SERVER-FILE-ERROR

A Guardian file-system operation on a server process failed unexpectedly or completed with an unexpected error code.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-SERVER-NAME token-type ZSPI-TYP-STRING.
ZSPI-TKN-MANAGER token-type ZSPI-TYP-FNAME32.
ZACL-TKN-EVENT-PROCEDURE token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-PROCESS-ID token-type ZSPI-TYP-PHANDLE.
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1038 - Server pathmon-name.serverclass-name file operation
procedure() failed, process process-id, file file-name, error
errnum
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has one of the values ZACL-VAL-EVENT-LEVEL-WARNING or ZACL-VALEVENT- LEVEL-ERROR. ZACL-TKN-EVENT-SERVER-NAME contains the name of the server class that contains the failed server process. This token is the subject of the message. ZSPI-TKN-MANAGER contains the name of the PATHMON process that manages the server class. ZACL-TKN-EVENT-PROCEDURE contains the name of the Guardian file-system procedure that reported the failure. ZACL-TKN-EVENT-PROCESS-ID contains the process handle of the server process accessed by the failed operation. If the process identified by this token still exists when the event message text is generated, the EMS event message formatting facilities convert process handle tokens to named or unnamed process descriptor strings. If the process no longer exists, EMS generates a question mark instead of the process descriptor string. ZACL-TKN-EVENT-FILE-NAME contains the process file name that identifies the server process and that was used by the failed operation. ZACL-TKN-EVENT-ERROR-CODE contains the file-system error code returned by the failed operation.

Effect. The Pathsend operation associated with the file-system operation is aborted.

Recovery. Correct the condition that caused the operation to fail.

EMS Messages 1040 through 1049

1042: ZACL-EVT-SERVER-UNAUTH-MSG

A Pathsend requestor process attempted to perform a Pathsend operation on a server class that the process is not authorized to communicate with. Authority to communicate with a Pathway server class is determined by the OWNER and SECURITY configuration attributes of the server class.

Unconditional Tokens
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL. ZACL-TKN-EVENT-SERVER-NAME token-type ZSPI-TYP-STRING. ZSPI-TKN-MANAGER token-type ZSPI-TYP-FNAME32. ZACL-TKN-EVENT-ACCESS-ID token-type ZSPI-TYP-UINT.

Table Continued

Event-Message Text

```
1042 - Unauthorized attempt to invoke server pathmonname.
serverclass-name, access-user access-id
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-SERVER-NAME contains the name of the server class specified in the Pathsend operation. This token is the subject of the event message. ZSPI-TKN-MANAGER contains the name of the PATHMON process that manages the server class. ZACL-TKN-EVENT-ACCESS-ID contains the process access user identifier of the unauthorized requestor process.

Effect. The Pathsend operation fails with an FESERVERLINKCONNECT Pathsend error and FESECVIOL file-system error detail.

Recovery. None. This event message indicates a possible attempt to breach the security of a Pathway application.

EMS Messages 1050 through 1059

TS/MP 2.6 or later does not contain messages in this number range.

EMS Messages 1060 through 1069

1062: ZACL-EVT-NO-BACKUP-CPU-AVAIL

The backup CPU specified in the domain-coordinator process startup string is not available or no backup CPU was specified, and no other CPUs are available on the system.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1062 - ( process-name): No backup CPU available
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-PROCESS-NAME contains the name of the domain coordinator process followed by the CPU and PIN enclosed in parentheses. This token is the subject of the event message.

Effect. The domain coordinator process runs as a single process.

Recovery. If a ZACL-EVT-BACKUP-CPU-DISABLED event occurred, then use the CONTROL command to restart the ACS subsystem processes in the CPU. If that does not correct the problem, then reload the failed CPUs.

1063: ZACL-EVT-INVALID-BACKUP-CPU

An invalid CPU number was specified in the domain coordinator process startup string.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1063 - Invalid backup CPU specified: cpu-number
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL. ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT. contains the invalid CPU number. This token is the subject of the event message.

Effect. The primary domain coordinator process terminates immediately.

Recovery. Start the domain coordinator process. Specify a valid CPU for the backup process or specify the value -1 to allow the domain coordinator process to select a backup CPU.

1064: ZACL-EVT-ACS-PROCNAME-IN-USE

One of the process names reserved for ACS subsystem processes is currently in use by another process.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1064 - ACS reserved process name process-name is in use.
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-ERROR. ZACL-TKN-EVENT-PROCESS-NAME contains the reserved process name that is currently being used. This token is the subject of the event message.

Effect. If the domain coordinator process detects the problem during process startup, it terminates immediately. If it detects the condition during the `START ACS` command processing, the START operation fails and no ACS subsystem processes are started.

Recovery. Stop the process that uses the ACS reserved process name.

1065: ZACL-EVT-CORRUPT-BC-INI-FILE

The ACS subsystem domain-coordinator process-initialization file (BCINI) is invalid or contains invalid data.

Unconditional Tokens

`ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.`
`ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.`
 	
Event-Message Text
 	
`1065 - Corrupt or invalid BCINI file: file-name`

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-FILE-NAME contains the Guardian name of the domain-coordinator process-initialization file (BCINI). This token is the subject of the event message.

Effect. The domain coordinator cannot save the current ACS subsystem configuration to the BCINI file.

Recovery. Use the `STOP` command to stop the ACS subsystem, and then purge the corrupt file and start the subsystem again.

1066: ZACL-EVT-BACKUP-CPU-DISABLED

A primary domain coordinator process terminated unexpectedly in the indicated CPU and the ACS subsystem processes in that CPU have not been restarted.

Unconditional Tokens
`ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.` `ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.`

Table Continued

Event-Message Text

```
1066 - Invalid backup CPU cpu-number - CPU needs to be
d
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-CPU-NUMBER contains the CPU number. This token is the subject of the event message.

Effect. The primary domain coordinator continues to run as a single process if a backup process was configured for a specific backup CPU. If the STARTUPMSG attribute specified a value of -1, the domain coordinator attempts to start a backup process in the next available CPU.

Recovery. Use the `CONTROL ACS` command to restart the ACS subsystem processes in the indicated CPU.

1067: ZACL-EVT-INVALID-ACS-PROD-PATH

The domain coordinator program does not reside in the current subvolume used to load the system ($SYSTEM.SYSnn).

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-SYSNN token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1067 - Invalid ACS product path actual-subvol - expecting
system-subvol
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-FATAL. ZACL-TKN-EVENT-FILE-NAME contains the name of the Guardian subvolume where the domain coordinator program resides. This token is the subject of the event message. ZACL-TKN-EVENT-SYSNN contains the name of the Guardian subvolume that contains the OSIMAGE file used to load the system.

Effect. The domain coordinator process terminates immediately.

Recovery. Contact Hewlett Packard Enterprise support to install the ACS program files in $SYSTEM.SYSnn.

1068: ZACL-EVT-INVALID-HOMETERM

The home terminal is no longer valid and cannot be used to create a backup domain coordinator process or other ACS subsystem process.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-FILE-NAME token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1068 - Hometerm file-name is no longer valid for process
process-name
```

Token Descriptions

`ZACL-TKN-EVENT-LEVEL`

has the value of ZACL-VAL-EVENT-LEVEL-ERROR.

`ZACL-TKN-EVENT-FILE-NAME`

contains the name of the home terminal. This token is the subject of the event message.

`ZACL-TKN-EVENT-PROCESS-NAME`

contains the name of the process that was to be created with the invalid home terminal.

Effect. If the backup domain coordinator process fails, the primary domain coordinator process continues to run without a backup process. If one of the other ACS subsystem processes fails, the CPU where it was running is disabled.

Recovery. Restart the home terminal if possible. Otherwise, stop and restart the ACS subsystem with a new home terminal.

1069: ZACL-EVT-PROCESS-SWITCH

The primary process can no longer continue.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1069 - Process pair for process-name is switching over
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-INFO. ZACL-TKN-EVENT-PROCESS-NAME contains the name of the process pair.

Effect. Primary process terminates and the backup process takes over.

Recovery. Informational message only; no corrective action is needed.

EMS Messages 1070 through 1071

1070: ZACL-EVT-PROCESS-SWITCH-FAILED

The process pair cannot switch because no backup process is running. The backup process CPU might be down or the ACS subsystem processes have not been restarted in that CPU.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-PROCESS-NAME token-type ZSPI-TYP-STRING.
```

Event-Message Text

```
1070 - Process pair for process-name cannot switch over
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-INFO. ZACL-TKN-EVENT-PROCESS-NAME contains the name of the process pair.

Effect. The process pair cannot switch; ACS subsystem processes might not be available to handle requests.

Recovery. If the backup-process CPU is down, then reload it. If the backup process CPU is up, then use the CONTROL command to restart the ACS subsystem processes.

1071: ZACL-EVT-PRIMARY-ABENDED

The primary domain coordinator process terminated abnormally.

Unconditional Tokens

```
ZACL-TKN-EVENT-LEVEL token-type ZACL-TYP-EVENT-LEVEL.
ZACL-TKN-EVENT-CPU-NUMBER token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-CODE token-type ZSPI-TYP-INT.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT.
```

Table Continued

Conditional Tokens

```
ZACL-TKN-EVENT-TERM-STOPPER token-type ZSPI-TYP-PHANDLE.
ZACL-TKN-EVENT-TERM-SSID token-type ZSPI-TYP-SSID.
ZACL-TKN-EVENT-TERM-TEXT token-type ZSPI-TYP-STRING.
ZACL-TKN-EVENT-ERROR-DETAIL token-type ZSPI-TYP-INT.
```

Event-Message Text

```
1071 - Primary process in CPU cpu-number terminated
abnormally, comp-code comp-code, stopper stopper-id, termssid
term-ssid, term-info term-info, term-text term-text
```

Token Descriptions

ZACL-TKN-EVENT-LEVEL
has the value of ZACL-VAL-EVENT-LEVEL-WARNING. ZACL-TKN-EVENT-CPU-NUMBER contains the number of the CPU on the local system in which the primary domain coordinator process was running. This token is the subject of the event message. ZACL-TKN-EVENT-ERROR-CODE contains the completion code associated with the process-termination operation. A completion code of -4 indicates that only the primary process, not both members of the process pair, terminated. ZACL-TKN-EVENT-ERROR-DETAIL contains the termination information code associated with the process termination. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message. If the process was stopped externally (indicated by completion code 6), the value of this token is the creator user identifier of the external process that initiated the termination. ZACL-TKN-EVENT-TERM-STOPPER contains the process handle of the process that stopped the primary process. This token is included only if the process was stopped externally. ZACL-TKN-EVENT-TERM-SSID contains the SPI subsystem identifier associated with the primary process termination. This token is omitted if no subsystem identifier was associated with the process termination. ZACL-TKN-EVENT-TERM-TEXT contains the termination text associated with the primary process termination. This token is omitted if no termination text was associated with the process termination.

Table Continued

`ZACL-TKN-EVENT-ERROR-DETAIL`

contains the termination information code associated with the process termination. If the value of this token is 0 (zero), the token is null and can be regarded as omitted from the message. If the process was stopped externally (indicated by completion code 6), the value of this token is the creator user identifier of the external process that initiated the termination.

Effect. The backup domain coordinator process becomes the primary domain coordinator process and creates a new backup process.

Recovery. If the problem persists, contact your Hewlett Packard Enterprise support analyst.

Websites

General websites

Hewlett Packard Enterprise Information Library

www.hpe.com/info/EIL

Hewlett Packard Enterprise Support Center

www.hpe.com/support/hpesc

Contact Hewlett Packard Enterprise Worldwide

www.hpe.com/assistance

Subscription Service/Support Alerts

www.hpe.com/support/e-updates

Software Depot

www.hpe.com/support/softwaredepot

Customer Self Repair

www.hpe.com/support/selfrepair

Manuals for L-series

http://www.hpe.com/info/nonstop-ldocs

Manuals for J-series

http://www.hpe.com/info/nonstop-jdocs

For additional websites, see **Support and other resources**.

Support and other resources

Accessing Hewlett Packard Enterprise Support

- For live assistance, go to the Contact Hewlett Packard Enterprise Worldwide website:

  http://www.hpe.com/assistance

- To access documentation and support services, go to the Hewlett Packard Enterprise Support Center website:

  http://www.hpe.com/support/hpesc

Information to collect

- Technical support registration number (if applicable)
- Product name, model or version, and serial number
- Operating system name and version
- Firmware version
- Error messages
- Product-specific reports and logs
- Add-on products or components
- Third-party products or components

Accessing updates

- Some software products provide a mechanism for accessing software updates through the product interface. Review your product documentation to identify the recommended software update method.
- To download product updates:

  Hewlett Packard Enterprise Support Center

  www.hpe.com/support/hpesc

  Hewlett Packard Enterprise Support Center: Software downloads

  www.hpe.com/support/downloads

  Software Depot

  www.hpe.com/support/softwaredepot

- To subscribe to eNewsletters and alerts:

  www.hpe.com/support/e-updates

- To view and update your entitlements, and to link your contracts and warranties with your profile, go to the Hewlett Packard Enterprise Support Center **More Information on Access to Support Materials** page:

www.hpe.com/support/AccessToSupportMaterials

IMPORTANT: Access to some updates might require product entitlement when accessed through the Hewlett Packard Enterprise Support Center. You must have an HPE Passport set up with relevant entitlements.

Customer self repair

Hewlett Packard Enterprise customer self repair (CSR) programs allow you to repair your product. If a CSR part needs to be replaced, it will be shipped directly to you so that you can install it at your convenience. Some parts do not qualify for CSR. Your Hewlett Packard Enterprise authorized service provider will determine whether a repair can be accomplished by CSR.

For more information about CSR, contact your local service provider or go to the CSR website:

http://www.hpe.com/support/selfrepair

Remote support

Remote support is available with supported devices as part of your warranty or contractual support agreement. It provides intelligent event diagnosis, and automatic, secure submission of hardware event notifications to Hewlett Packard Enterprise, which will initiate a fast and accurate resolution based on your product's service level. Hewlett Packard Enterprise strongly recommends that you register your device for remote support.

If your product includes additional remote support details, use search to locate that information.

Remote support and Proactive Care information

HPE Get Connected

www.hpe.com/services/getconnected

HPE Proactive Care services

www.hpe.com/services/proactivecare

HPE Proactive Care service: Supported products list

www.hpe.com/services/proactivecaresupportedproducts

HPE Proactive Care advanced service: Supported products list

www.hpe.com/services/proactivecareadvancedsupportedproducts

Proactive Care customer information

Proactive Care central

www.hpe.com/services/proactivecarecentral

Proactive Care service activation

www.hpe.com/services/proactivecarecentralgetstarted

Warranty information

To view the warranty for your product or to view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products* reference document, go to the Enterprise Safety and Compliance website:

www.hpe.com/support/Safety-Compliance-EnterpriseProducts

Additional warranty information

HPE ProLiant and x86 Servers and Options

www.hpe.com/support/ProLiantServers-Warranties

HPE Enterprise Servers

www.hpe.com/support/EnterpriseServers-Warranties

HPE Storage Products

www.hpe.com/support/Storage-Warranties

HPE Networking Products

www.hpe.com/support/Networking-Warranties

Regulatory information

To view the regulatory information for your product, view the *Safety and Compliance Information for Server, Storage, Power, Networking, and Rack Products*, available at the Hewlett Packard Enterprise Support Center:

www.hpe.com/support/Safety-Compliance-EnterpriseProducts

Additional regulatory information

Hewlett Packard Enterprise is committed to providing our customers with information about the chemical substances in our products as needed to comply with legal requirements such as REACH (Regulation EC No 1907/2006 of the European Parliament and the Council). A chemical information report for this product can be found at:

www.hpe.com/info/reach

For Hewlett Packard Enterprise product environmental and safety information and compliance data, including RoHS and REACH, see:

www.hpe.com/info/ecodata

For Hewlett Packard Enterprise environmental information, including company programs, product recycling, and energy efficiency, see:

www.hpe.com/info/environment

Documentation feedback

Hewlett Packard Enterprise is committed to providing documentation that meets your needs. To help us improve the documentation, send any errors, suggestions, or comments to Documentation Feedback (**docsfeedback@hpe.com**). When submitting your feedback, include the document title, part number, edition, and publication date located on the front cover of the document. For online help content, include the product name, product version, help edition, and publication date located on the legal notices page.

Syntax Summary

This appendix contains summaries of the PATHCOM commands described in this manual. All commands are listed in alphabetic order. Note that PATHCOM commands that create or manage Pathway/iTS objects such as TCP, TERM, and PROGRAM objects, are described in *Pathway/iTS System Management Manual*.

NOTE: PDMCOM commands in TS/MP 2.6 or later follow the same syntax as PATHCOM commands. For more information on PDMCOM Commands, see the *TS/MP 2.7 ACS Reference Manual*.

```
ADD SERVER server-class [ , server-attribute ]...
```

```
ALTER [ SERVER ] server-class
   { , server-attribute                                    }
   { , DELETE delete-option                                }
   { , RESET server-keyword                                }
   { , RESET (server-keyword [ , server-keyword ]...)      }

delete-option is:
   [ ASSIGN logical-unit  ]
   [ DEFINE define-name   ]
   [ PARAM parameter-name ]
   [ PROCESS process-name ]
   [ STARTUP              ]
```

```
CMDCWD oss-pathname
```

```
CMDVOL { [ \node. | \*.] $volume.subvolume }
       { [ \node. | \*.] $volume           }
       { [ \node. | \*.] subvolume         }
       { \node | \*                        }
```

```
CONTROL PATHMON , pathmon-attribute [ , pathmon-attribute
]...

pathmon-attribute is:
     BACKUPCPU number
     DUMP { ON [ ( FILE file-name ) ] | OFF }
     DUMPMEMORY (FILE file-name )
            SPREBALANCECPU number
```

```
DELETE [SERVER] { server-class                                    }
                { ( server-class [ , server-class ]... ) }
```

```
ERRORS [ number ]
```

```
EXIT
```

```
FC [ [-] number | string ]
```

```
FREEZE { [ SERVER ] server-class                          }
       { [ SERVER ] (server-class [, server-class ]...) }
       { SERVER *                                         }
   [ ! ]
   [ , WAIT ]
```

```
HELP [ / OUT list-file / ] [ ALL           ]
                           [ COMMANDS      ]
                           [ command-name  ]
                           [ "detail-name" ]
                           [ error-num     ]
```

```
HISTORY [ number ]
```

```
INFO [ / OUT list-file / ] PATHMON [ , OBEYFORM ]
```

```
INFO [ / OUT list-file / ] PATHWAY [ , OBEYFORM ]
```

```
INFO [ / OUT list-file / ]
    { [ SERVER ] server-class }
    { [ SERVER ] ( server-class [ , server-class ] ...) }
    { SERVER * }
    [ , OBEYFORM ]
```

```
LOG1 [ file-name [ , logparam [ , logparam ] ] ]
LOG2 [ file-name [ , logparam [ , logparam ] ] ]
logparam is:
     STATUS
     EVENTFORMAT
```

```
{ OBEY } file-name
{ O    }
```

```
OBEYVOL [ [ \node.] $volume.subvolume ]
        [ [ \node.] $volume           ]
        [ [ \node.] subvolume         ]
        [ \node                       ]
```

```
OPEN [ \node.]$pm-process
```

```
PATHCOM [ / run-option [ , run-option ]... /        ]
        [ pathmon-name [ ; command [ ; command ]... ] ]

run-option is:

        IN command-file
        OUT list-file
        CPU number
        NAME $process-name
        NOWAIT
        PRI number
```

```
PATHMON / NAME $process-name [ , run-option ]... /

run-option is:
        CPU number
        INSPECT ON
        NOWAIT
        OUT log1-file
        PRI number
```

```
PRIMARY PATHMON [ , IFPRICPU number ]
```

```
RESET CMDCWD
```

```
RESET SERVER [ server-keyword [ , server-keyword ]... ]

server-keyword is:

    ARGLIST          GUARDIAN-SWAP       PRI
    ASSIGN           HIGHPIN             PROCESS
    AUTORESTART      HOMETERM            PROCESSTYPE
    CPUS             IN                  PROGRAM
    CREATEDELAY      LINKDEPTH           SECURITY
    CWD              MAXLINKS            STARTUP
    DEBUG            MAXSERVERS          STDERR
    DEFINE           NUMSTATIC           STDIN
    DELETEDELAY      OUT                 STDOUT
    ENV              OWNER               TMF
    GUARDIAN-LIB     PARAM               VOLUME
```

```
SET PATHMON , pathmon-attribute [ , pathmon-attribute ]...

pathmon-attribute is:

    BACKUPCPU number
    DUMP { ON [ ( FILE file-name ) ] | OFF }
```

```
SET PATHWAY pw-attribute [ , pw-attribute ]...

pw-attribute is:

    DUMASK Octalnumber
    MAXASSIGNS number
    MAXPARAMS number
    MAXSERVERCLASSES number
    MAXSERVERPROCESSES number
    MAXSTARTUPS number
    MAXTCPS number
    MAXTERMS number
    MAXDEFINES number
    MAXEXTERNALTCPS number
    MAXLINKMONS number
    MAXPATHCOMS number
    MAXPROGRAMS number
    MAXSPI number
    MAXTELLQUEUE number
    MAXTELLS number
    MAXTMFRESTARTS number
    NODEINDEPENDENT [ON | OFF]
    OWNER ownerid
    SECURITY security-attribute
    SPREBALANCEMODE rbmodeval
```

NOTE: Among the attributes in the `SET PATHWAY` command are some attributes that apply only to the Pathway/iTS product: MAXTCPS, MAXTERMS, MAXEXTERNALTCPS, MAXPROGRAMS, MAXTELLQUEUE, MAXTELLS, and MAXTMFRESTARTS. Values for these Pathway-specific attributes must be included in your configuration file, even if the Pathway/iTS product is not installed at your site. The syntax diagram lists all the SET PATHWAY attributes, including those for the Pathway/iTS product. For full descriptions for Pathway/iTS-only attributes, see the *Pathway/iTS System Management Manual*.

```
SET SERVER server-attribute [ , server-attribute ]...

server-attribute is:
     PROCESSTYPE { GUARDIAN | OSS }
     ARGLIST argument [,...]
     ASSIGN logical-unit , assign-spec
     AUTORESTART number
     CPUS { ( primary:backup [, primary:backup ]...) |
          ( cpu [ , cpu ]... )
          ( cpu (cpu-wt) [ , cpu (cpu-wt) ]... )
     CREATEDELAY number { HRS | MINS | SECS | CSECS }
     CWD oss-pathname
     DEBUG { ON | OFF }
     ( DEFINE define-name , define-attribute-spec
          [, define-attribute-spec ] ... )
     DELETEDELAY number { HRS | MINS | SECS }
     ENV name=value [ ,... ]
     GUARDIAN-LIB file-name
     HIGHPIN { ON | OFF }
     HOMETERM file-name
     IN file-name
     LIKE server-class
     LINKDEPTH number
     MAXLINKS number
     MAXSERVERS number
     NUMSTATIC number
     OUT file-name
     OWNER ownerid
     PARAM parameter-name parameter-value [ , ... ]
     PRI priority
     PROCESS $process-name [ ( process-attribute , ... ) ]
     PROGRAM { file-name | oss-pathname }
     SECURITY security-attribute
     STARTUP string
     STDERR oss-pathname
     STDIN oss-pathname
     STDOUT oss-pathname
     TIMEOUT number { HRS | MINS | SECS }
     TMF { ON | OFF }
     UMASK octalnumber
     VOLUME volume-spec

logical-unit is:
     { [ program-unit . ] } logical-file-name
     { * .               }

assign-spec is:

     [ file-name                        ]
     [ [ file-name ] , create-spec ... ]
create-spec is:
     { EXT [ ( ] pri-extent-size [ ) ]                }
     { EXT ( [ pri-extent-size ] , sec-extent-size )  }
```

```
     { EXCLUSIVE                                          }
     { SHARED                                             }
     { PROTECTED                                          }
     { I/O                                                }
     { INPUT                                              }
     { OUTPUT                                             }
     { CODE file-code                                     }

process-attribute is:

     [ ASSOCIATIVE { ON | OFF}   ]
     [ CPUS primary:backup | cpu ]
     [ DEBUG { ON | OFF }        ]
     [ GUARDIAN-SWAP $volume     ]
     [ HOMETERM file-name        ]
     [ PRI number                ]
```

```
SHOW [ / OUT list-file / ] { CMDCWD }
     { CMDVOL }
     { ERRORS }
     { OBEYVOL }
```

```
SHOW [ / OUT list-file / ] SERVER
```

```
SHUTDOWN [ ! ] [ , WAIT ]
```

```
SHUTDOWN2 [, MODE   { OR[DERLY] | AB[ORT] | IM[MEDIATE]   } ]
          [, STATUS { QU[IET] | AG[GREGATE]               } ]
          [, UNTIL  { DONE                                }
                    { TIMEOUT number { HRS | MINS | SECS }} ]
```

```
START PATHWAY { COOL } [!]
              { COLD }
```

```
START
   { [ SERVER ] server-class [ , PROCESS $process-name ] }
   { [ SERVER ] ( server-class [ , server-class ]... ) }
   { SERVER * }
```

```
STATS [ / OUT list-file / ]
   { [ SERVER ] server-class }
   { [ SERVER ] ( server-class [ , server-class ]... ) }
   { SERVER * }

   [ , server-attribute [ , server-attribute ]... ]

server-attribute is:

   COUNT number
   FREQTABLE
   INTERVAL number { HRS | MINS | SECS }
```

```
STATUS [ / OUT list-file / ]
  { [ LINKMON ] L\node.$process-name }
  { LINKMON * }
```

```
STATUS [ / OUT list-file / ] PATHMON
```

```
STATUS [ / OUT list-file /                        ] PATHWAY
       [ , COUNT number                           ]
       [ , INTERVAL number { HRS | MINS | SECS } ]
```

```
STATUS [ / OUT list-file / ]
   { [ SERVER ] name [ , PROCESS $process-name ]        }
   { [ SERVER ] name [ , PROCESSES             ]        }
   { [ SERVER ] name [ , DETAIL                ]        }
   { [ SERVER ] name [ , FREEZE                ]        }
   { [ SERVER ] ( name [ , name ]... ) [ , PROCESSES ]  }
   { [ SERVER ] ( name [ , name ]... ) [ , DETAIL    ]  }
   { [ SERVER ] ( name [ , name ]... ) [ , FREEZE    ]  }
   { SERVER * [ , PROCESSES ] [ , DETAIL ] [ , FREEZE ] }
   { FREEZE }
```

```
STOP PATHMON
```

```
STOP { [ SERVER ] server-class                          }
     { [ SERVER ] ( server-class [ , server-class ]... ) }
     { SERVER *                                         }
```

```
SWITCH PATHMON
```

```
THAW { [ SERVER ] server-class                              }
     { [ SERVER ] ( server-class [, server-class ]... ) }
     { SERVER *                                             }
```

```
! [ [-]number | string ]
```

PATHCOM Reserved Words

This appendix contains a list of words that are reserved and cannot be used for variable names in PATHCOM commands. Although it might be possible in some contexts to use words on this list, Hewlett Packard Enterprise strongly recommends that these words not be used in any user-assigned names.

NOTE: PDMCOM commands in TS/MP 2.6 or later follow the same syntax as PATHCOM commands. For more information on PDMCOM Commands, see the *TS/MP 2.7 ACS Reference Manual*.

NonStop TS/MP Environment

The reserved words in the NonStop TS/MP environment appear in the following table:

ADD ALTER ALL ARGLIST ASSIGN ASSOCIATIVE AUTORESTART AUTO	BACKUP BACKUPCPU BLOCK BOTH
CMDCWD CMDVOL CODE COLD COMMANDS CONTROL CONVERSATIONAL COOL COUNT COUPLE CPUS CREATEDELAY CSECS CWD	DEBUG DISABLE DECOUPLE DEFINE DELETE DELETE-DELAY DETAIL DOMAIN DUMASK DUMP DUMPMEMORY
ENV ERRORS EVENTFORMAT EXCLUSIVE EXIT EXT	FC FILE FREEZE FREQTABLE
GUARDIAN GUARDIAN-LIB GUARDIAN-SWAP	HELP HIGHPIN HOMETERM HRS

Table Continued

I-O IBM-3270 IFPRICPU IN INFO INPUT INTELLIGENT INTERVAL	LIKE LINKDEPTH LINKMON LOG1 LOG2
MANUAL MAXASSIGNS MAXDEFINES MAXEXTERNALTCPS MAXLINKS MAXLINKMONS MAXPARAMS MAXPATHCOMS MAXPROGRAMS MAXSERVERCLASSES MAXSERVERS MAXSPI MAXSTARTUPS MAXTCPS MAXTELLQUEUE MAXTELLS MAXTERMS MAXTMFRESTARTS	NUMSTATIC NODEINDEPENDENT
OBEY OBEYFORM OBEYVOL OFF ON OPEN OUT OUTPUT OWNER	PARAM PARTITION PATHMON PATHWAY POWERONRECOVERY PRI PRIMARY PROCESS PROCESSES PROCESSTYPE PROGRAM PROTECTED

Table Continued

REC RESET	SECS SECURITY SEL SERVER SET SHARED SHOW SHUTDOWN SHUTDOWN2 SPREBALANCECPU SPREBALANCEMODE START STARTUP STATS STATUS STDERR STDIN STDOUT STOP SWITCH
T16-6510 T16-6520 T16-6530 T16-6530WP T16-6540 THAW TIMEOUT TMF	UMASK UNDO UNTIL
VOLUME	WAIT WARM

Pathway/iTS Environment

Reserved words exclusive to the Pathway/iTS environment appear in the following table:

ABORT	BREAK
CHECK-DIRECTORY CODEAREALEN CURRENT	DIAGNOSTIC DISPLAY-PAGES
ECHO ERROR-ABORT	INITIAL INSPECT IOPROTOCAL IS-ATTACHED

Table Continued

MAXINPUTMSGLEN MAXINPUTMSGS MAXPATHWAYS MAXREPLY MAXSERVERPROCESSES MAXTERMDATA	NonStop
OSS	PRINTER
REFRESH-CODE RESUME RUN	SERVERPOOL STATE SUSPEND SWAP
TCLPROG TCP TELL TERM TERMBUFF TERMPOOL TRAILINGBLANKS TYPE	

Configuration Limits and Defaults

The following table lists global limits for various items within each Pathway environment.

NOTE: These limits are subject to change with product SPRs or with new software releases. For information on TS/MP 2.6 or later limits, see the *TS/MP 2.7 ACS Reference Manual*.

Table 13: Global Pathway Environment Limits

Item	Limits Per Pathway Environment
ASSIGNs	8191 for all server classes.
Components	Internal table limit of 64 KB for total of named components (server classes, server processes, ACS subsystem processes, ASSIGNs, PARAMs, and Pathway/iTS TCPs, TELL objects, and TERM objects) within a Pathway environment.
	Internal table limit of 8 KB for each of the following within a Pathway environment: server classes, server processes, ASSIGNs, PARAMs, DEFINEs, and Pathway/iTS TCPs, TELL objects, and TERM objects.
LINKMON processes	One per processor.
	PATHMON processes: Maximum of 256 Pathway environments with which a ACS subsystem process can communicate.
	Server classes: Maximum of 1024 server classes to which a ACS subsystem process can have outstanding links across all Pathway environments.
	Links to server processes: Maximum of 1750 concurrent links to server processes across all Pathway environments, per ACS subsystem process.
	Maximum concurrent links to the same server process, per ACS subsystem process, is the value of LINKDEPTH.
	Concurrent Pathsend operations: Maximum of 1600 concurrent, outstanding Pathsend calls per ACS subsystem process.
	Pathsend requestors: Maximum of 1024 concurrent Pathsend requestors per ACS subsystem process.

Table Continued

Item	Limits Per Pathway Environment
Links	Only one SCREEN COBOL program at a time can use a link (by performing a SEND).
	Only one Pathsend process at a time can use a link.
	For a description of how to calculate the number of links a link manager might require for a given server class and the number of links a given server class is able to provide, see **Configuring Links for Optimum Performance** on page 67.
PATHMON process	One (or one process pair) per Pathway environment.
PATHWAY object	One (or one process pair) per Pathway environment.
Requestors	Maximum number of concurrently running processes (ACS subsystem, PATHCOM, SPI, and Pathway/iTS TCP and external TCP,) must not exceed 800. (Although more than 150 can be configured, no more than 800 can run concurrently.)
	Maximum number of requests from a requestor to the same server process is 255 or the value of LINKDEPTH, whichever is smaller.
Pathsend requestors	Maximum of 255 concurrent, outstanding send requests per Pathsend requestor.
SERVER objects	Maximum of 4095 supported by each PATHMON process; suggested maximum of 4094 to avoid error 1102.
	For information on the number of links to a server class, see the discussion earlier on links.

The following table lists each PATHCOM parameter that has a value range or limit, its corresponding SPI token or field name, its value range, and the 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 Pathway environments. For more information on this interface, see the *TS/MP 2.5 Management Programming Manual*.

Table 14: Limits and Defaults for Parameters

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
ARGLIST	ZPWY-TKN-DEFSCARGLIST	Values: OSS process startup argument list; from 2 to 24,000 characters. ZPWY-TKN-DEF-SCENV + ZPWY-TKN-DEF-SCARGLIST cannot exceed 24,000 bytes Default: Empty argument list
ASSOCIATIVE	ZASSOCIATIVE	Values: ON | OFF Default: OFF
AUTORESTART	ZAUTORESTART	Values: 0 through 32,767Value must be greater than 0. Default: 0
BACKUPCPU	ZCPUZCPUPAIR	Values: 0 through 15. Value must be a processor that is greater than plus or minus 1 from the primary processor. Default: Depends on command
COUNT	N.A.	Values: 0 through 65,536 Default: 1
CPU CPUS	ZCPU ZCPUPAIR	Values: For processor pairs, maximum of 16 pairs, 0 through 15.Backup processor numbers need not be unique.For a sequence of single CPUs, maximum of 16 CPUs, 0 through 15. Each processor must be unique. Default: Determined by the PATHMON process (all available CPUs are used).
CREATEDELAY	ZCREATEDELAY	Values: HRS | MINS | SECS | CSECS 0 through 16,383 CSECS, or 0 through 16,383 SECS, or 0 through 1092 MINS, or 0 through 18 HRS The time set for this parameter can be added to the response time for all transactions; therefore, the value must be small. Default: 1 MINS

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
CWD	ZPWY-TKN-DEFSCCWD	Values: Absolute OSS pathname of the current working directory. Default: None
DEBUG	ZDEBUG	Values: ON | OFF Default: OFF
DELETEDELAY	ZDELETEDELAY	Values: HRS | MINS | SECS 0 through 16,383 SECS, or 0 through 1092 MINS, or 0 through 18 HRS Default: 10 MINS
ENV	ZPWY-TKN-DEFSCENV	Values: OSS process environment variable list.; from 3 to 24,000 characters.ZPWY-TKN-DEF-SCENV + ZPWYTKN- DEF-SCARGLIST cannot exceed 24,000 bytes. Default: Empty environment variable list
ERRORS	N.A	Values: • -1: Allow infinite number of errors. • 0: Stop after encountering first error. • 132,767: Number of errors and warnings to be ignored. Default: 0
EVENTFORMAT	ZEVENTFORMAT	Values: EVENTFORMAT Default: Only text (console) messages are generated.
EXT	ZEXTENTSIZE	Values: 1 through 65,535 Default: No value is passed.
FILE	ZFILE	Values: One through seven alphanumeric characters, including the $; first character after the $ must be alphabetic. Default: N.A.
GUARDIAN-SWAP		Values: Maximum of 10 disk volumes for SERVER objects. Default: Selected by operating system (from _DEFAULTS DEFINE statement, if available).

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
HELP	N.A.	Values: 1000 through 3999 Default: N.A.
HOMETERM	ZHOMETERM	Values: Name of home terminal for servers in class. Default: Home terminal of the PATHMON process.
IN	ZIN	Values: Name of input file. Default: Spaces are passed for file name.
LINKDEPTH	ZLINKDEPTH	Values: 1 through 255 Value can be less than or equal to the value of MAXLINKS. If MAXLINKS is 0, use the LINKDEPTH default. Default: 1
MAXASSIGNS	ZMAXASSIGNS	Values: 0 through 8191 Default: N.A.
MAXDEFINES	ZMAXDEFINES	Values: 0 through 8191 Default: N.A
MAXEXTERNALTCPS	ZMAXEXTERNALTCPS	Total of values for MAXTCPS, MAXEXTERNALTCPS, and MAXLINKMONS must not exceed 800. Maximum number of concurrently running processes (TCP, external TCP, LINKMON, PATHCOM, and SPI) must not exceed 800. Recommended value if Pathway/iTS is not installed at your site: 0
MAXLINKMONS	ZMAXLINKMONS	Values: Total of values for MAXTCPS, MAXEXTERNALTCPS, and MAXLINKMONS must not exceed 800. Maximum number of concurrently running processes (TCP, external TCP, ACS subsystem, PATHCOM, and SPI) must not exceed 800. Default: 0

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
MAXLINKS	ZMAXLINKS	Values: 0 through 4095 To help determine the appropriate value for this attribute, see the discussion on Links earlier in this appendix. In a COBOL server, the size of MAXLINKS size must be equal to or less than the value of RECEIVETABLE. Hewlett Packard Enterprise recommends that the size suggested of RECEIVE-TABLE be larger than MAXLINKS so that you can increase the configuration without recompiling. Default: Unlimited number of links To help determine the appropriate
MAXPARAMS	ZMAXPARAMS	Values: 0 through 4095 The value of MAXPARAMS must not exceed value of MAXSERVERPROCESSES. Default: N.A.
MAXPATHCOMS	ZMAXPATHCOMS	Values: 1 through 100 Maximum number of concurrent openers of the PATHMON process (TCP, external TCP, ACS subsystem, PATHCOM, and SPI) must not exceed 800. For production, the value for this parameter must be large enough to allow for parallel interfaces (multiple PATHCOM and SPI processes). Default: 5
MAXPROGRAMS	ZMAXPROGRAMS	Values: 0 through 4095 Default: 0
MAXSERVERCLASSES	ZMAXSERVERCLASSES	Values: 0 through 4095 Total of MAXSERVERS values for all server classes must not exceed value of MAXSERVERPROCESSES. Default: N.A.
MAXSERVERPROCESSES	ZMAXSERVERPROCESSES	Values: 0 through 4095 Default: N.A.

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
MAXSERVERS	ZMAXSERVERS	Values: 0 through 4095 Total of MAXSERVERS values for all server classes must not exceed value of MAXSERVERPROCESSES. Default: 1
MAXSPI	ZMAXSPI	Values: 1 through 100 Maximum number of concurrent openers of the PATHMON process (TCP, external TCP, ACS subsystem, PATHCOM, and SPI) must not exceed 800. For production, the value for this parameter must be large enough to allow for parallel interfaces (multiple PATHCOM and SPI processes). Default: 1
MAXSTARTUPS	ZMAXSTARTUPS	Values: 0 through 4095 Value must not exceed value of MAXSERVERCLASSES. Default: 1
MAXTCPS	ZMAXTCPS	Values: Total of values for MAXTCPS, MAXEXTERNALTCPS, and MAXLINKMONS must not exceed 800. Maximum number of concurrent openers of the PATHMON process (TCP, external TCP, LINKMON, PATHCOM, and SPI) must not exceed 800. Recommended value if Pathway/iTS is not installed at your site: 0.
MAXTELLQUEUE	ZMAXTELLQUEUE	Values: 0 through 300 Recommended value if Pathway/iTS is not installed at your site: 0
MAXTELLS	ZMAXTELLS	Values: 0 through 4095 Recommended value if Pathway/iTS is not installed at your site: 0
MAXTERMS	ZMAXTERMS	Values: 0 through 4095 Recommended value if Pathway/iTS is not installed at your site: 0

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
MAXTMFRESTARTS	ZMAXTMFRESTARTS	Values: -1: Unlimited restarts 0: No restarts 1 through 32,767 Number of restarts Default: 5
MODE	ZMODE	Values: ORDERLY | ABORTTERM | IMMEDIATE Default: ORDERLY
N.A.	ZACCESS	Values: I/O | INPUT | OUTPUT Default: No value is passed.
N.A.	ZEXCLUSION	EXCLUSIVE | SHARED| PROTECTED Default: No value is passed.
N.A.	ZFILECODE	0 through 65,535 Default: 0
N.A.	ZRECORDSIZE	1 through 4072 Default: No value is passed.
N.A.	ZBLOCKSIZE	Values: 1 through 4096 Default: No value is passed.
NODEINDEPENDENT	ZNODEINDEPEN DENT	Values: ON | OFF Default: OFF
N.A.	ZNODEINDEPEN DENTFORMAT	Values: YES | NO NO
NUMSTATIC	ZNUMSTATIC	Values: 0 through 4095; must not exceed value of MAXSERVERS. The number of dynamic servers for a server class is MAXSERVERS minus NUMSTATIC. For better performance, the value must be close to or equal to MAXSERVERS. Default: 0 (all servers are dynamic)

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
OUT	ZOUT	Values: Name of output file. Default (SERVER): Spaces are passed for file name. Default (PATHCOM): Output directed to PATHCOM list file (typically the home terminal).
OWNERID	ZOWNERID	Values: [system-number.] group-number, user-number or [nodename.] groupname. user-name Default: Owner ID of user who started the PATHMON process.
PRI	ZPRIORITY	Values: 1 through 199 Default (the PATHMON process): The priority assigned by $CMON, or if $CMON is not active, the priority of TACL minus 1. Default (SERVER): 10 less than the priority of the PATHMON process. Default (Pathway/iTS TCP): 20 less than the priority of the PATHMON process.
PROCESS	ZLINKMON ZPROCESS	Values: One to six alphanumeric characters, including the $; first character after the $ must be alphabetic. Default: N.A.
PROCESSTYPE	ZPROCESSTYPE	Values: GUARDIAN | OSS Default: GUARDIAN
PROGRAM	ZPROGRAM	Values: For Guardian server processes, the name of the Guardian object file name; 1 through 15 alphanumeric or hyphen characters; first character must be alphabetic.
	ZPWY-TKN-DEFSCPROGRAMOSS	For OSS server processes, the OSS pathname of the object file name; maximum of 1024 characters. Default: None

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
SECURITY	ZSECURITY	Values: “A” “G” “O” “-” “N” “C” “U” Any local userA group member or ownerOwner onlyLocal super IDAny local or remote userAny member of owner’s communityAny member of owner’s user class. The owner must not be a supergroup user with “N” or “A” for a security value; this combination causes a security breach. Default: “O”
SPREBALANCEMODE	ZSPREBALANCEMODE	Values: “DISABLE”, ”AUTO”, “MANUAL” Default: “DISABLE”
STATUS	ZSTATUS	Values: STATUS Default: Only error messages are written to log file.
STATUS	N.A.	Values: QUIET | AGGREGATE Default: QUIET
STDERR	ZPWY-TKN-DEFSCSTDERR	Values: Name of the OSS stderr (standard error) file.Must not exceed 1024 bytes. Default: None
STDIN	ZPWY-TKN-DEFSCSTDIN	Values: Name of the OSS stdin (standard input) file. Must not exceed 1024 bytes. Default: None
STDOUT	ZPWY-TKN-DEFSCSTDOUT	Values: Name of the OSS stdout (standard output) file. Must not exceed 1024 bytes. Default: None
TIMEOUT	ZTIMEOUT	Values: 0 through 16,383 SECS, or 0 through 1092 MINS, or 0 through 18 HRS
		Default (LINKMON): The LINKMON process waits indefinitely for server I/O operation to complete unless the send times out during the I/O because of the timeout value specified in the SERVERCLASS_SEND_ call.
		Default (Pathway/iTS TCP): No timeout; TCP waits indefinitely for server I/O operation to complete.

Table Continued

PATHCOM Parameter*	SPI Token/Field**	Value Limits and Defaults
TMF	ZTMF	Values: ON | OFF Default (SERVER): OFF Pathway/iTS Default (PROGRAM): ON Pathway/iTS Default (TERM): ON
UNTIL	N.A.	Values: DONE | TIMEOUT Default: DONE
*PATHCOM is the interactive management interface.		
**SPI is the DSM management programming interface.		

Migration Information

This appendix covers migration and compatibility issues for the NonStop TS/MP product on C-series and D-series systems. These topics are discussed:

- Interprocess communication issues
- Application conversion

For information on migration and compatibility issues relative to the Pathway/iTS product, see the *Pathway/iTS System Management Manual*.

For more information about running applications on NonStop systems, 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 NonStop systems. In addition, interoperability between PATHMON environment processes on C-series systems and D40 or later NonStop systems 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 NonStop systems, see the “Migration Information” appendix in the D30.02 version of the **NonStop TS/MP and Pathway System Management Guide**.

NOTE: For information on migrating applications from TS/MP 2.0 to TS/MP 2.6 or later, see TS/MP 2.7 ACS Reference Manual.

Interprocess Communication Issues

The NonStop TS/MP product supports D-series operating system features as described in this list.

- Requestor processes can run at high and low PINs.
- Server processes can run at a high PIN under the following conditions:
  - The Pathway `HIGHPIN` option is set to ON for the 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 server 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 SERVER` command.

If you have configured your system 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. The following table lists the possible communication paths for D-series Pathway 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 15: NonStop TS/MP Process High-PIN and Low-PIN Support

Operation	PATHMON	PATHCOM	LINKMON
Run at a high PIN	Yes	Yes	No
Service high-PIN server	Yes	N.A.	Yes
Service high-PIN requestors	Yes, when requestors are named	N.A.	Yes
Open a high-PIN process	Yes	Yes	Yes
Create a high-PIN process	Yes	N.A.	N.A.
Can be created by a high-PIN process	Yes	Yes	N.A.

By default, PATHCOM runs at a high PIN.

By default, the PATHMON process runs at a high PIN. The PATHMON process can create server processes at high PINs, if the following conditions are met:

- The `HIGHPIN` option is set to ON for the object. (The default value is OFF).
- The `HIGHPIN` option is d 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 requestor 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 servers must follow the C-series representation of a dollar sign ($) followed by one to four alphanumeric characters.
- Process names for PATHCOM and the PATHMON process must follow the Cseries representation as follows:

- If the name will be used across a network, a dollar sign ($) followed by one to four alphanumeric characters.
- If the name will not be used across a network, a dollar sign ($) followed by one to five 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.

If your environment includes Pathway/iTS, see the *Pathway/iTS System Management Manual*.

For more information about converting your applications, see the *Guardian Application Conversion Guide*.

Setting TMF Parameters

When you are configuring and controlling a PATHMON application that uses the Transaction Management Facility (TMF) subsystem of the TMF, consider these basic questions:

- How do the settings you specify for the *TMF* parameter of the `SET SERVER` command affect Pathsend procedure calls and SCREEN COBOL SEND statements?
- 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 requestor 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 SERVER Command and TMF

The `SET SERVER` command contains a *TMF* parameter with an `ON` or `OFF` option. By setting this parameter to one of these, you control the types of operations a server class can perform:

- TMF ON--The ACS subsystem process or TCP allows a Pathsend server-class send operation or SCREEN COBOL SEND to members of this server class whether or not the Pathsend or SCREEN COBOL program is in transaction mode.
- TMF OFF—The ACS subsystem process or TCP allows a Pathsend server-class send operation or SCREEN COBOL SEND to the members of this server class only if the Pathsend or SCREEN COBOL program is not in transaction mode. OFF is the default setting.

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. An application that needs TMF to provide database consistency is at risk when TMF is not running.

To determine how to address the TMF error, see the **HPE NonStop TMF Operations and Recovery Guide**.

Glossary

NOTE: This glossary does not include terms for elements of the SCREEN COBOL language that are also found in standard COBOL. For definitions of such terms, refer to standard COBOL texts or to the text of the Pathway/TS SCREEN COBOL Reference Manual.

$RECEIVE

A special Guardian file name through which a process receives and optionally replies to messages from other processes by using Guardian procedure calls. This file is analogous to a request queue defined for a NonStop TUXEDO server.

absolute pathname

An OSS pathname that begins with a slash (/) character and is resolved beginning with the root directory. See also OSS pathname, relative pathname, and root directory.

accept operation

An operation in which a screen program waits for a response from the terminal and allows data to be input into the program data area from the terminal.

advisory message

A message displayed in the terminal advisory field to inform the terminal operator of errors detected during input checking.

API

See application program interface (API).

application

A complete set of programs or routines that perform a function. See also Pathway application, NonStop TUXEDO application, and PTP application.

Application Cluster Services (ACS) subsystem

An infrastructure used by transaction processing middleware products to manage and access server classes.

Application Cluster Services (ACS) domain

A defined set of objects (resources) within a single physical NonStop system that are controlled and managed as a unit by the ACS subsystem.

application program interface (API)

A set of services (such as programming language functions or procedures) that are called by an application program to communicate with other software components. For example, an API might consist of a set of procedure calls that provide a workstation application with a standard interface for communicating with a Tandem system. Other examples of APIs are the ATMI in Novell TUXEDO systems and NonStop TUXEDO systems, the Customer Information Control System (CICS) command-level API, and the Pathsend procedures.

application terminal

A terminal on which a Pathway application runs. See also command terminal.

Application-Transaction Monitor Interface (ATMI)

The application programming interface to the System/T transaction monitor in a NonStop TUXEDO system. This interface includes transaction routines, message handling routines, service interface routines, and buffer-management routines.

assignment

The use of an ASSIGN command to make logical file assignments for programs in the Guardian environment. A logical assignment equates a Tandem file name with a logical file of a program and, optionally, attributes characteristics to that file.

associative server

A process within a server class that can be started outside the Pathway environment by a process other than the PATHMON process that controls the server class.

ATMI

See Application-Transaction Monitor Interface (ATMI).

attributes

Those characteristics of an object that influence the operation of that object and establish its capabilities.

audit trail

A record of database changes that can be used by the TMF subsystem to rebuild a database in the event of a hardware or software failure. An audit trail is also known in the industry as a transaction log.

audited file

A database file that is flagged for auditing by the TMF subsystem; auditing is the monitoring of transactions in preparation for recovery efforts.

availability

The amount of time an application running on a Tandem system can be used effectively by a user of that application.

backup process

The member of a process pair that takes over the application work when the primary process fails. See also primary process, process pair, and checkpoint message.

base screen

In SCREEN COBOL, a screen that occupies the entire physical display area of a terminal and can be displayed independently of other screens. This type of screen can contain areas on which overlay screens are displayed. See also screen and overlay screen.

batch processing

A method of transaction processing in which transactions are first grouped together and then processed at regular intervals. See also online transaction processing (OLTP).

block mode

A terminal operating mode in which data is read from the terminal and displayed on the terminal one screen at a time. See also conversational mode.

cache

A temporary storage buffer.

cascading server

A term formerly used for a nested server. See nested server.

checkpoint message

In the Guardian environment, a message sent by a primary process to the backup process that keeps the backup process up to date on the state of the application. A checkpoint message provides a

snapshot of process activity that can be used in the event of a takeover by a backup process to allow the backup process to maintain fault-tolerant operation.

CICS

See Customer Information Control System (CICS).

CISC

See complex instruction-set computing (CISC).

client

An application program that requests services to be performed. In discussions of the Pathway environment, this term is used to refer to the part of an application that runs on some other vendor's hardware, such as a personal computer, Macintosh computer, UNIX workstation, or mainframe computer system, and makes requests of a server process. See also requestor, server, and client/server model.

client/server model

A model for distributing applications. In general, but not always, in this model the client process resides on a workstation and the server process resides on a second workstation, minicomputer, or mainframe system. Communication takes the form of request and reply pairs, which are initiated by the client and serviced by the server. (A server can make requests of another server, thus acting as a client.) Client/server computing is often used to connect different types of workstations or personal computers to a host computer system by means of supported communications protocols. See also requestor/server model.

client/transaction server model

A model for client/server applications. The client/transaction server model is the model of choice for high-volume OLTP applications in which transaction volume is great and the processing requirements change infrequently.

In the Tandem environment, an application following this model divides processing between a client running on a workstation and servers running on a Tandem system. The client handles the user interface and business logic and processing. The servers store information for use by the client and handle database input and output functions. Interprocess communication (IPC) messages transfer data between client and server.

COBOL

The Tandem compiler and run-time support for the American National Standards Institute (ANSI) programming language COBOL, X.3.23-1985. Pathway server processes are often written in this language.

cold start

The operation that starts a PATHMON environment for the first time. This operation either creates a new PATHMON configuration file (PATHCTL file) that defines the PATHMON environment and its objects or overwrites an existing PATHMON configuration file (which effectively creates a new PATHMON environment). See also cool start.

command file

A file that serves as a source for command input. For example, users can prepare a command file containing PATHCOM or SCREEN COBOL Utility Program (SCUP) commands. They can then cause the commands in the file to be executed by issuing the PATHCOM or SCUP OBEY command and specifying the name of the file. Alternatively, they can specify this file as the input file when they execute PATHCOM or SCUP.

command interpreter

An interactive program used to run programs, check system status, create and delete disk files, and alter hardware states.

command terminal

A terminal at which a system manager or operator enters commands for configuration and management, such as the PATHCOM commands that configure and manage a PATHMON environment. See also application terminal.

complex instruction-set computing (CISC)

A processor architecture based on a large instruction set, characterized by numerous addressing modes, multicycle machine instructions, and many special-purpose instructions. See also reduced instruction-set computing (RISC).

configuration

The definition or alteration of characteristics of an object. See also object.

configuration subscriber (CS)

A process that executes in each processor within a system and publishes critical ACS subsystem configuration information to the other ACS subsystem processes.

configured TERM object

A TERM object that is explicitly configured with an ADD TERM command. Such a TERM object exists until it is explicitly deleted. Names of configured TERM objects begin with a letter. See also temporary TERM object and TERM object.

consistency

See database consistency.

context

Information required by a server to process the current request in an exchange of multiple request and reply messages: for example, identification of the last item processed. See also context-free server and terminal context.

context-free server

A server that does not retain any information about the processing of previous requests. A context-free server accepts a single message from a requestor, performs the requested tasks, and issues a single reply to respond to the requestor. After the reply message is issued, the server retains no information, or context, that can be used in subsequent requests. Pathway servers must be context-free. A context-free server is analogous to a NonStop TUXEDO request/response server. Tandem subsystems are context-free servers; therefore, management applications using the Subsystem Programmatic Interface (SPI) to communicate with Tandem subsystems must pass back context information in continuation requests. See also context and Subsystem Programmatic Interface (SPI).

conversational mode

A terminal operating mode in which data is read from the terminal and displayed on the terminal screen one line at a time. See also block mode and intelligent mode.

cool start

The operation that restarts a PATHMON environment, using the information in an existing PATHMON configuration file (PATHCTL file). The PATHMON environment is previously started with a cold start operation. See also cold start.

Crossref cross-reference generator

A Tandem software tool that produces a cross-referenced listing of selected identifiers such as data variables, statement labels, or subprograms in an application program.

current working directory

The OSS directory from which relative pathnames are resolved. See also OSS pathname and relative pathname.

Customer Information Control System (CICS)

An IBM transaction management system that provides concurrent online access to data files by means of user-written application programs. CICS also includes facilities for building, using, and maintaining databases.

Data Definition Language (DDL)

1. The set of data definition statements within the Structured Query Language (SQL).
2. A Tandem product for defining data objects in Enscribe files and translating object definitions into source code.

data integrity

The condition of a database when the data values are accurate, valid, and consistent according to rules established for changing the database. See also database consistency.

database consistency

The state of a database in which items satisfy established criteria. For example, an account balance must equal credits to the balance minus debits to the balance. When the database satisfies these criteria, the database is considered to be consistent. In general, a database is consistent when it is accurate and all changes generated by transactions are complete. Database consistency is defined by the application, which establishes the values and relationships of database fields and records.

database management system (DBMS)

A product, such as NonStop SQL/MP or Enscribe, that serves as the interface between a user or program (for example, a Pathway server) and the database. Among its many functions, the DBMS controls access to and organization of data within the database.

DBCS

See double-byte character set (DBCS).

DDL

See Data Definition Language (DDL).

deadlock

1. A situation in which two processes cannot proceed because each is waiting for a reply from the other.
2. A situation in which two transactions cannot proceed because each is waiting for the other to release a lock.

dedicated device

A term formerly used for a terminal or other input/output device controlled by a configured TERM object, so that a Pathway application always ran on that device without having to be started from PATHCOM with a RUN PROGRAM command. (No new term replaces this term; instead, the manual

text now refers to such devices as those associated with configured TERM objects). See also nondedicated device and configured TERM object.

default value

The value that the system uses for a particular attribute or parameter when a value has not been supplied by the user.

DEFINE

A named set of attributes and associated values. In a `DEFINE` (as with an `ASSIGN` command), users can specify information to be communicated to processes they start.

definition files

A set of files containing data declarations for items related to SPI messages and their processing. The core definitions required to use SPI are provided in a DDL file and in several language-specific definition files, one for each programming language that supports SPI. The Tandem DDL compiler generates the language-specific files from the DDL file. Subsystems that support SPI provide additional definition files containing subsystem-specific definitions.

delimiters

Characters that make it possible for a SCREEN COBOL requestor and an external device or front-end process to exchange compact variable-length messages efficiently; delimiters can be message delimiters or field delimiters.

descriptor

For each elementary data item, the SCREEN COBOL compiler builds a data structure that describes the size, type, usage, and dependencies of the item. All of the information that pertains to a given item makes up the descriptor for that item. For example, the PICTURE specification is included in the descriptor. The descriptors are passed to the TCP in the pseudocode and provide a dictionary of information for interpreting and handling incoming data. When the MAP or SMAP compiler option is used, the descriptors appear in the compiler map at the end of the listing.

diagnostic screen

A screen of information that is displayed to inform the terminal operator of error conditions and termination status.

disk process

In the Tandem environment, the portion of the operating-system software that performs read, write, and lock operations on disk volumes and creates TMF audit records. See also file system.

display attribute

A terminal display feature that is given a screen data name. The screen data name can be associated with a predefined system name in the SPECIAL-NAMES paragraph and thus be manipulated by a SCREEN COBOL program.

distributed data

Information (for example, customer names and addresses, inventory items, and personnel records) that resides on more than one node in a network and can be accessed by authorized users from any node in that network.

distributed processing

A type of processing environment in which resources are distributed among CPUs within a single system or spread across a network of systems. A user on any network node can, if properly authorized, access resources and database files anywhere within the network.

Distributed Systems Management (DSM)

A group of tools for managing a variety of subsystems in a distributed processing environment.

distributed transaction processing (DTP)

The coordination of transactions among application servers residing within an Expand network and possibly accessing different database management systems (NonStop SQL/MP and Enscribe). DTP allows the coordination of multiple, autonomous actions as a single logical unit of work.

double-byte character

A character represented in two bytes. See also double-byte character set.

double-byte character set (DBCS)

A character set, such as Tandem Kanji, that uses two bytes of data to represent a single character.

DSM

See Distributed Systems Management (DSM).

dumb terminal

See fixed-function terminal.

dynamic server

A server process that the PATHMON process creates after a TCP or ACS subsystem process has waited for a specified time period for a static server to become available. A dynamic server process exists only as long as it is needed. See also static server.

EDIT file

A source text file that can be augmented and modified by the user through a Tandem text editor program such as TEDIT (PS Text Edit).

EMS

See Event Management Service (EMS).

EMS log file

See event log file.

Enscribe database record manager

Tandem database management software that provides a record-at-a-time interface between servers and a distributed database. See also NonStop SQL/MP.

event log file

A file maintained by the Event Management Service (EMS) for logging of event messages.

Event Management Service (EMS)

A part of DSM used to provide event collection, event logging, and event distribution facilities. It provides for different descriptions of events for people and for programs, lets an operator or application select specific event message data, and allows for flexible distribution of event messages within a system or network. EMS has an SPI-based programmatic interface for reporting and retrieving events. See also event message.

event message

A special kind of SPI message that describes an event occurring in the system or network. Event messages are collected, logged, and distributed by EMS. See also Event Management Service (EMS).

Expand networking software

Tandem software that can connect up to 255 Tandem NonStop systems into a single network.

expandability

See scalability.

Extended General Device Support (GDSX)

A Tandem product that facilitates communication between general I/O devices and a PATHMON environment by acting as a front-end or a back-end process.

extensible structured token

In the Subsystem Programmatic Interface (SPI), a token with a value that can be extended by appending new fields in later releases. The token is accessed through reference to a token map containing field-version and null-value information, allowing SPI to provide compatibility between different versions of the structure. See also simple token and token (definition 2).

external PATHMON process

See external process.

external process

A process in a different PATHMON environment from the process with which it is communicating. For example, suppose a TCP managed by PATHMON process $PMB requests a link to a server process in a server class that is managed by PATHMON process $PMA. Both the TCP and PATHMON process $PMB are external processes with respect to PATHMON process $PMA and the server class managed by $PMA.

external server

See external process.

external TCP

See external process.

fault tolerance

The ability of a Tandem NonStop system to continue processing despite the failure of any single software or hardware component within the system.

field-characteristic clause

In SCREEN COBOL, an ordered set of characters that specify the characteristics of a screen field.

file identifier (fileID)

In the Guardian environment, the portion of a file name following the subvolume name. In the OSS environment, a portion of the internal information used to identify a file in the OSS file system. The two identifiers are not comparable.

file name

In the Guardian environment, the set of node name, volume name, subvolume name, and file identifier characters that uniquely identifies a file. This name is used to open a file and thereby provide a connection between the opening process and the file. See also fully qualified file name, partially qualified file name, OSS filename, and OSS pathname.

file system

1. In the Guardian environment, the application program interface for communication between a process and a file. A file can be a disk file, a device other than a disk, or another process.
2. In the OSS environment, a collection of files and file attributes. A file system provides the namespace for the file serial numbers that uniquely identify its files.

File Utility Program (FUP)

A Tandem product that allows users to create, copy, purge, and otherwise manipulate disk files interactively.

file-name expansion

The expansion of a partially qualified Guardian file name for a disk file to include the associated node, volume, and subvolume names.

fixed-function terminal

A nonintelligent device (that is, a device without processing ability) capable of sending and receiving information over communications lines. Fixed-function terminals are often referred to as dumb terminals.

freeze condition

A condition in which communication between a terminal and a server class is prohibited. See also thaw condition.

front-end process

A process, such as the process that accepts the TCP terminal-data stream, that serves as the intermediary between one system, process, or device and another.

fully qualified file name

The complete name of a file in the Guardian environment. For a permanent disk file, this consists of a node name (system name), volume name, subvolume name, and file identifier (file ID). In interactive interfaces such as PATHCOM and TACL, the parts of a file name are separated by periods. See also partially qualified file name.

gateway process

A process, such as the Transaction Delivery Process (TDP) that is part of RSC, that manages communications between dissimilar environments (for example, a workstation and a Tandem system). A gateway process both transfers information and converts it to a form compatible with the protocols used by the destination environment.

GDSX

See Extended General Device Support (GDSX).

graphical user interface (GUI)

A type of screen interface that typically includes pull-down menus, icons, dialog boxes, and online help.

Guardian

An environment available for interactive or programmatic use with the Tandem NonStop Kernel. Processes that run in the Guardian environment use the Guardian system procedure calls as their application program interface, and might also use related APIs such as the Pathsend and TMF procedure calls. See also Open System Services (OSS).

Guardian environment

The Guardian API, tools, and utilities. See also Guardian.

Guardian operating environment

See Guardian environment.

high PIN

A process identification number (PIN) in the range 256 through 65535. See also process identification number (PIN) and low PIN.

I/O process

In the Guardian environment, a system process to manage input/output devices. Applications use the Guardian file system to send requests to I/O processes. See also file system.

IDS

See intelligent device support (IDS) facility.

Inspect

The Tandem debugging tool that can be used interactively to examine and modify the execution of Guardian processes and SCREEN COBOL requestor programs.

Inspect command terminal

The terminal on which programmers enter commands to Inspect when debugging a SCREEN COBOL program or a Pathway server.

intelligent device

A device such as an automatic teller machine, a point-of-sale device, or a communications line, or a process such as a Guardian process, that can communicate with the Pathway environment through the intelligent device support (IDS) facility, the Remote Server Call (RSC) product, or the Pathsend procedure calls.

intelligent device support (IDS) facility intelligent device support (IDS) facility

A feature of the TCP that supports access to Pathway server classes by intelligent devices. This facility allows SCREEN COBOL requestor programs to interact with external processes that, in turn, control devices such as personal computers, automated teller machines, and point-of-sale devices.

intelligent mode

An operating mode in which data and messages are sent between an intelligent device and the Pathway environment. See also conversational mode (definition 1), intelligent device, and message-oriented requestor.

interactive mode

An operating mode in which commands are entered from a terminal keyboard.

interoperability

The ability to communicate, execute programs, or transfer data between dissimilar environments.

interoperate

To communicate, execute programs, or transfer data between dissimilar environments.

interprocess communication (IPC) message

The unit of communication between requestors and servers. An IPC message consists of a request message and a reply message.

IPC message

See interprocess communication (IPC) message.

keyword

A word in a command string or programming language that must be spelled and positioned in a prescribed way, usually to indicate the meaning of an adjacent parameter.

library

A set of related files or common files that can be accessed by multiple programs or processes.

linear expandability

See scalability.

link

- An open of a server process within a server class. When a link manager—that is, a TCP or a ACS subsystem process—sends a request to a PATHMON process for a link to a server in a specified server class, the PATHMON process selects a server process in that server class (possibly starting a new server process if necessary) and then returns the name of the server process to the requesting link manager. See also link granting and link manager.
- To examine, collect, associate together, and modify code and data blocks from one or more object files to produce a target object file. On Tandem NonStop systems, linking for TNS/R native object files is performed by the `nld` utility.

link access

The actual transfer of data from a requestor to a server process. In the Pathway environment, link access is provided by TCPs and ACS subsystem processes. See also link granting.

link granting

The process of selecting a particular server process in a server class to handle a request from a link manager (TCP or ACS subsystem process) on behalf of a requestor. In the Pathway environment, link granting is done by the PATHMON process. See also link and link access.

link management

The act of coordinating the sharing of links between requestor or client processes and server processes.

link manager

A process that requests links to server processes and provides link access after the link is granted. TCPs and ACS subsystem processes are the link managers in the Pathway environment. See also link access and link granting.

LINKMON process

A Guardian process that supports access to servers in the Pathway, NonStop TUXEDO, and PTP environments. LINKMON processes act as link managers for requestors and clients that use the Pathsend procedure calls and other interfaces such as RSC, POET, and the NonStop TUXEDO ATMI. See also link manager.

lock

A mechanism that coordinates access to the same data; locks are either shared or exclusive.

log file

See PATHMON log file and event log file.

low PIN

A process identification number (PIN) in the range 0 through 255. (PIN 255 is reserved by the system; it is never assigned to a running process, but is used by high-PIN processes to communicate with low-PIN processes.) See also process identification number (PIN) and high PIN.

MAKEUL

A TACL macro used to perform pTAL compilation of user-written conversion routines for use with the Pathway/TS TCP and SCREEN COBOL requestors and to create the TNS/R native TCP user library containing these routines.

manageability

The ability to easily and comprehensively manage a subsystem, system, or network.

management application

An application program that automates configuration and management tasks. Such a program can request from the PATHMON process the same kinds of services that system managers can request through the PATHCOM interface. A management application can also interact with subsystems other than the Pathway subsystem. Management applications use the Subsystem Programmatic Interface (SPI) to send commands to subsystems and the Event Management Service (EMS) to receive notification of significant events.

Message Section

A section in the Data Division of a SCREEN COBOL source program that describes the data type, size, and relative position (sequence) of each field in a message template. This section also defines the editing and conversion that must be performed on each field. See also message-oriented requestor.

message-oriented requestor

A SCREEN COBOL requestor that sends data from working storage to a device (or to a front end process that controls a device) and receives data from the device or process into working storage by way of Message Section templates. SCREEN COBOL requestors that use the intelligent device support (IDS) facility are message-oriented. These requestors use SEND MESSAGE statements and their REPLY clauses in the Procedure Division to interact with the intelligent devices or front-end processes. See also screen-oriented requestor.

mixed data item

A data item that contains both single-byte and double-byte characters; in a COBOL or SCREEN COBOL program, these data items are declared as PIC X.

modified data tag (MDT)

In SCREEN COBOL, a bit that is set or reset to indicate whether data in an associated field is to be sent to the computer from the terminal.

multithreaded

A programming model that provides more than one thread of control within a program. Multithreading allows multiple sequential processing tasks to be executed concurrently within a process: for example, a terminal control process (TCP). See also thread and single-threaded.

native System/T client

In the NonStop TUXEDO environment, a client program that executes in the Open System Services (OSS) environment and communicates directly with System/T. An example of this type of client is the Pathway translation server for the NonStop TUXEDO system. Also called a NonStop TUXEDO native client.

nested server

A server that acts as a requestor by sending requests to other servers. In the Pathway environment, such requests are made by calls to Pathsend procedures.

nld utility

A utility that collects, links, and modifies code and data blocks from one or more object files to produce a target TNS/R native object file.

no-early-reply rule

The rule that states that when a server process reads a request message, it must completely process the request before it replies to it.

node

A Tandem NonStop system that is part of an Expand network. The name of the node, also called the system name, is the first of four parts of a file name in the Guardian environment. See also Expand networking software.

nondedicated device

A term formerly used for a terminal or other input/output device on which a Pathway application could be started from PATHCOM with a `RUN PROGRAM` command. The `RUN PROGRAM` command results in the creation of a temporary TERM object to control the terminal. (No new term replaces this term; instead, the manual text now refers to such devices as those associated with temporary TERM objects.) See also dedicated device and temporary TERM object.

noninteractive mode

An operating mode in which commands are entered through a command file.

NonStop Kernel

See Tandem NonStop Kernel.

NonStop processing

On Tandem NonStop systems, processing characterized by continued operation even when a component fails, when equipment is being repaired or replaced, or while new processors or peripheral devices are being added to the system. In the Guardian environment, NonStop processing is provided by means of fault tolerance and process pairs.

NonStop SQL/MP

The Tandem relational database management system that promotes efficient online access to large distributed databases. See also Structured Query Language (SQL) and Enscribe database record manager.

NonStop Transaction Manager/MP (NonStop TM/MP)

A Tandem software product that provides transaction management, transaction protection, and database consistency in online transaction processing (OLTP) environments. It gives full protection to transactions that access NonStop SQL/MP and Enscribe databases, as well as recovery capabilities for transactions, online disk volumes, and entire databases. The component of NonStop TM/MP that provides these features is the TMF subsystem. See also Transaction Management Facility (TMF) subsystem.

NonStop Transaction Services/MP (NonStop TS/MP)

A Tandem product that provides process management and link management functions for OLTP applications on Tandem NonStop systems. NonStop TS/MP consists of the PATHMON process, the ACS subsystem process, the PATHCOM process and interface, and the Pathsend procedures. Together with NonStop Transaction Manager/MP (NonStop TM/MP), NonStop TS/MP forms the foundation for Tandem's open transaction processing services, including those provided by the RSC

and POET products, the NonStop TUXEDO system, and the Parallel Transaction Processing (PTP) Services for the CICS API product. See also Pathway/TS and NonStop Transaction Manager/MP (NonStop TM/MP).

NonStop TUXEDO application

See NonStop TUXEDO transaction processing environment.

NonStop TUXEDO native client

See native System/T client.

NonStop TUXEDO server

A server process or program managed by the NonStop TUXEDO system administrative facilities. See also Pathway server.

NonStop TUXEDO system

Tandem's implementation of the Novell, Inc., TUXEDO enterprise transaction processing system. Major features of the NonStop TUXEDO implementation include the use of the Tandem core services (Tandem NonStop Kernel, NonStop Transaction Services/MP, and NonStop Transaction Manager/MP) to provide the Tandem fundamentals (scalability, availability, and manageability) for TUXEDO applications. The NonStop TUXEDO system runs in the OSS operating environment on Tandem NonStop systems. See also OSS environment.

NonStop TUXEDO transaction processing environment

An environment that provides the API and transaction monitor functions of the Novell, Inc., TUXEDO transaction processing system in addition to the benefits provided by the Tandem core services: Tandem NonStop Kernel, NonStop Transaction Services/MP, and NonStop Transaction Manager/MP. See also Pathway transaction processing environment.

OBEY command

A command in a Tandem interactive interface, such as PATHCOM or the SCREEN COBOL Utility Program (SCUP), that allows users to execute the commands in a command file. See also command file.

object

An entity that is subject to independent reference or control by one or more subsystems. Examples of objects are devices, communications lines, processes, and files. In the PATHMON environment, the types of objects referred to or controlled by PATHCOM are PATHMON, PATHWAY, LINKMON, SERVER, TCP, TERM, PROGRAM, and TELL. The PATHWAY object, used with the `SET PATHWAY` and `STATUS PATHWAY` commands, refers to an entire PATHMON environment. The LINKMON object is only referred to, not controlled, by PATHCOM: that is, users can use PATHCOM to obtain information about LINKMON processes but cannot use it to make any changes in the configuration or state of those processes. See also PATHMON environment.

object attributes

See attributes.

object file

A file generated by a compiler, linker, or binder that contains machine instructions and other information needed to construct the executable code spaces and initial data for a process. The file can be a complete program that is ready for immediate execution, or it can be incomplete and require linking with other object files before execution. Compilers for languages such as COBOL produce object code. See also pseudocode file.

object type

In the Subsystem Programmatic Interface (SPI) or in an interactive interface such as PATHCOM, a category of objects to which a specific object belongs. Object types for PATHCOM include PATHMON, PATHWAY, LINKMON, SERVER, TCP, TERM, PROGRAM, and TELL. The SPI interface to the Pathway subsystem uses different names for some of these object types and defines additional object types; for example, PM, LM, PROGTERM, and TCPTERM are object types in the SPI interface.

OLTP

See online transaction processing (OLTP).

OLTP application

See online transaction processing (OLTP) application.

online transaction processing (OLTP)

A method of processing transactions in which entered transactions are immediately applied to the database. The information within the database is readily available to all users through online screens and printed reports. The transactions are processed while the requestor waits, as opposed to queued or batched transactions, which are processed at a later time. Online transaction processing can be used for many different kinds of business tasks such as order processing, inventory control, accounting functions, and banking operations. See also batch processing.

online transaction processing (OLTP) application

A set of programs that perform online transaction processing (OLTP) tasks on behalf of the user. With an OLTP application, many terminal users can update data simultaneously, recording the changes in the database as they are entered. OLTP applications generally display, check, and accept input data; manipulate the input data; and perform some type of data-output activity.

Open System Services (OSS)

An open system environment available for interactive or programmatic use with the Tandem NonStop Kernel. Processes that run in the OSS environment use the OSS application program interface; interactive users of the OSS environment use the OSS shell for their command interpreter. See also Guardian and Guardian environment.

OSS

See Open System Services (OSS).

OSS environment

The NonStop Kernel Open System Services (OSS) API, tools, and utilities. See also Open System Services (OSS).

OSS filename

A component of an OSS pathname containing any valid characters other than a slash (/) or a null. See also file name, OSS pathname, and file system (definition 2).

OSS operating environment

See OSS environment.

OSS pathname

The string of characters that uniquely identifies a file within its file system in the OSS environment. A pathname can be either absolute or relative. See also absolute pathname, relative pathname, and file system (definition 2).

OSS username

A string that uniquely identifies a user within the user database for a node.

overlay area

In SCREEN COBOL, an area of a base screen within which an overlay screen can be displayed.

overlay screen

In SCREEN COBOL, a screen that is displayed in an overlay area of a base screen. A base screen can be used with various overlay screens. See also screen and base screen.

Parallel Transaction Processing (PTP) Services for the CICS API product

The collection of software components (executable program files, data files, and so on) that support transaction processing and application development for CICS applications in the Guardian environment on Tandem NonStop systems. See also Customer Information Control System (CICS) and PTP transaction processing environment.

partially qualified file name

A Guardian file name in which only the right-hand file-name parts are specified. The remaining parts of the file name assume default values. See also fully qualified file name.

PATHCOM

- The interactive interface to the PATHMON process, through which users enter commands to configure and manage Pathway applications.
- The process that provides this interface.

PATHCOM command file

A file of PATHCOM commands that define and add the PATHMON-controlled objects required to execute an application. This file can contain all of the commands needed to start a PATHMON environment.

PATHCOM command terminal

See command terminal.

PATHCTL

See PATHMON configuration file.

Pathmaker product

A menu-driven application generator, provided by Tandem, that increases the productivity of programmers developing Pathway applications. The Pathmaker software generates requestor programs in SCREEN COBOL and server programs in C or COBOL.

PATHMON configuration file

A disk file in which a PATHMON process maintains configuration information for the objects under its control. The name of this file is PATHCTL.

PATHMON environment

The servers, server classes, TCPs, terminals, SCREEN COBOL programs, and tell messages that run together under the control of one PATHMON process.

PATHMON log file

A file used by a PATHMON process for reporting errors and changes in status.

PATHMON object

An object of type PATHMON; that is, a PATHMON process. See also PATHMON process and PATHMON-controlled object.

PATHMON process

The central controlling process in the Pathway environment. The PATHMON process maintains configuration-related data; grants links to server classes in response to requests from TCPs and ACS subsystem processes; and performs all process control (starting, monitoring, restarting, and stopping) of server processes and TCPs.

PATHMON-controlled object

An object defined and managed by a PATHMON process, through PATHCOM or the Pathway management programming interface. In the PATHCOM interface, such an object can be of type PATHWAY, PATHMON, SERVER, TCP, TERM, PROGRAM, or TELL. See also object and Pathway object.

pathname

See OSS pathname.

Pathsend procedures

The set of Guardian procedure calls that provide general access to Pathway server classes from any process on a Tandem system.

Pathsend process

A process, written as a Guardian program in C, C++, COBOL, Pascal, pTAL, or TAL, that makescalls to Pathsend procedures to request services from a Pathway server. A Pathsend process can be either a standard requestor, which initiates application requests, or a nested server, which is configured as a server class but acts as a requestor by making requests to other servers. A Pathsend process is also known as a Pathsend requestor.

Pathsend program

A Guardian program, written in C, C++, COBOL, Pascal, pTAL, or TAL, that makes calls to Pathsend procedures to request services from a Pathway server. A running Pathsend program is called a Pathsend process. See also Pathsend process.

Pathsend requestor

See Pathsend process.

PATHTCP2

The TCP object file, usually identified by the file name `$SYSTEM.SYSTEM.PATHTCP2`.

PATHTCPL

The TCP user library object file.

Pathway application

A set of programs that perform online transaction processing tasks in the Guardian environment, using interfaces defined by Tandem. A Pathway application can include SCREEN COBOL requestors, Pathsend requestors, and Pathway servers running on Tandem NonStop systems. It can also include GDSX front-end processes and clients that use RSC or POET. See also NonStop TUXEDO application and PTP application.

Pathway application development environment

A set of tools supporting the development of applications for the Pathway transaction processing environment. Depending on the customer’s needs and software configuration, this set of tools could include the SCREEN COBOL compiler and the SCREEN COBOL Utility Program (SCUP), the POET application development tools, the Pathmaker product, and various tools from Tandem Alliance partners. See also Pathway transaction processing environment.

Pathway Domain Management Interface (PDMI)

A management interface with a set of commands for configuring, controlling, and monitoring multiple PATHMON processes. Typically, it is used to work with multiple PATHMON processes under a single Pathway domain.

Pathway environment

See Pathway transaction processing environment.

Pathway management programming interface

A set of programmatic commands that allow users to write management application programs that communicate directly with the PATHMON process for configuration and management. This interface is based on the Subsystem Programmatic Interface (SPI) within the Distributed Systems Management (DSM) software. Programmatic commands communicating with the PATHMON process use the Pathway subsystem ID. See also Pathway subsystem and subsystem ID.

Pathway monitor process

See PATHMON process.

Pathway object

An object in the Pathway transaction processing environment. The set of Pathway objects is a more inclusive set than the set of PATHMON-controlled objects: for example, a ACS subsystem process is a Pathway object but not a PATHMON-controlled object. See also object, PATHMON-controlled object, and Pathway transaction processing environment.

Pathway Open Environment Toolkit (POET)

A set of programs and utilities that helps programmers create and run client/transaction server applications. In the POET environment, the client is a program running on a workstation, and the server is a Pathway server running on a Tandem NonStop system. POET uses the services of the Remote Server Call (RSC) product. The programming tools provided by POET include a simplified programming interface, name mapping, and conversion mapping.

Pathway server

A server process or program in the Pathway transaction processing environment. See also NonStop TUXEDO server and PTP server.

Pathway subsystem

The PATHMON environment components to which SPI commands are sent under the Pathway subsystem ID and which generate EMS event messages with the Pathway subsystem ID. All SPI commands for the Pathway subsystem are sent to the PATHMON process, but the processsing for the command might involve other processes, such as a TCP. Likewise, all EMS messages from the Pathway subsystem are generated by the PATHMON process, but the information might originate from another process. See also subsystem ID.

Pathway system

A term formerly used for the set of objects managed by a particular PATHMON process; now called PATHMON environment. See PATHMON environment.

Pathway transaction processing environment

A run-time environment consisting of Tandem's transaction-processing products for the Guardian operating environment. This term is often shortened to "Pathway environment." Depending on the customer's needs and software configuration, the Pathway environment could include NonStop TS/MP, the run-time portions of Pathway/TS (the TCP and the SCREEN COBOL run-time environment), NonStop TM/MP, GDSX processes, the run-time portion of the RSC product, the POET run-time environment, and the TRANSFER delivery system (when used as a workflow aid in transaction processing). See also NonStop TUXEDO transaction processing environment.

Pathway translation server for the NonStop TUXEDO system

A server process, provided by Tandem as part of the NonStop TUXEDO product, that allows a Pathway (SCREEN COBOL or Pathsend) requestor to use the services of a NonStop TUXEDO server. The translation server thus acts as a gateway process between the Pathway environment and the NonStop TUXEDO environment. Requestors that use this translation server must include special information in the header of each request message to identify the target NonStop TUXEDO application and service.

Pathway/TS

A Tandem product that provides tools for developing and interpreting screen programs to support OLTP applications in the Guardian environment on Tandem NonStop systems. Pathway/TS screen programs communicate with terminals and intelligent devices. Pathway/TS includes the TCP, the SCREEN COBOL compiler and run-time environment, and the SCREEN COBOL Utility Program (SCUP). It requires the services of the NonStop TS/MP product. See also NonStop Transaction Services/MP (NonStop TS/MP).

PDMCOM

A new command-line interface used to start PDMI, configure, and manage multiple Pathway environments at the same time. It is a superset of PATHCOM.

PIN

See process identification number (PIN).

POBJ

The default prefix, or file-name root, used by the SCREEN COBOL compiler in naming its output files. If no prefix is specified in the `RUN` command to run the compiler, the compiler produces a code file named POBJCOD, a directory file named POBJDIR, and (if the `SYMBOLS` option is d) a symbols file named POBJSYM.

POET

See Pathway Open Environment Toolkit (POET).

primary process

The currently active process of a process pair in the Guardian environment. See also backup process and process pair.

process

1. A unique execution of a program in the Guardian environment.
2. An entity in the OSS environment consisting of an address space, a single thread of control that executes within that address space, and the system resources required by that thread of control. See also process type.

process broker (PB) process

A process that executes in each processor within a system to manage the resources and link reservations for the processor.

NOTE: The PB process is part of the ACS subsystem.

process identification number (PIN)

An unsigned integer that identifies a process within a processor module in a Tandem NonStop system.

process management

The act of configuring, creating, and initializing processes; the monitoring and stopping of processes; and the recovery of failed processes. The PATHMON process provides process management functions for OLTP applications on Tandem NonStop systems.

process pair

A fault-tolerant arrangement of processes in the Guardian environment, whereby two processes in separate processors share the same name and execute identical code. One process functions as the primary process and the other functions as the backup process. The two processes are kept in sync through checkpoint messages sent from the primary to the backup process. If the primary process fails, the backup process is notified that it is now the primary, and it resumes the application work from the last valid checkpoint message.

process type

An attribute of a server class indicating whether the server processes in that server class are Guardian processes or OSS processes. See also process.

product

A tool provided by Tandem that allows users to develop simple data management applications without using a conventional programming language. The product can generate SCREEN COBOL programs for use with Enscribe databases.

program file

An executable object file. See also object file.

PROGRAM object

A template for creating and starting temporary TERM objects. See also temporary TERM object.

pseudo Pathsend procedure

In the Extended General Device Support (GDSX) software, one of the TSCODE-supported procedures that have Pathsend procedure counterparts.

pseudocode file

A file containing compiled code that is interpreted by software instead of being executed by the hardware. The SCREEN COBOL compiler produces pseudocode to be interpreted by the TCP. See also object file.

pTAL

A machine-independent systems programming language based on TAL. The pTAL language excludes architecture-specific TAL constructs and includes new constructs that replace the architecture-specific constructs. See also Transaction Application Language (TAL).

PTP application

A collection of software components that perform transaction processing for CICS applications in the Guardian environment on Tandem NonStop systems. See also Customer Information Control System (CICS) and Parallel Transaction Processing (PTP) Services for the CICS API product.

PTP server

A server in the PTP transaction processing environment. See also Customer Information Control System (CICS), PTP transaction processing environment, and Parallel Transaction Processing (PTP) Services for the CICS API product.

PTP Services product

See Parallel Transaction Processing (PTP) Services for the CICS API product.

PTP transaction processing environment

An environment that supports the CICS/ESA 3.3 command-level application program interface (API) and provides run-time services for CICS applications in the Guardian environment. The PTP environment provides the Tandem fundamentals for CICS applications. See also Customer Information Control System (CICS), Parallel Transaction Processing (PTP) Services for the CICS API product, and PTP application.

reduced instruction-set computing (RISC)

A processor architecture based on a relatively small and simple instruction set, a large number of general-purpose registers, and an optimized instruction pipeline that supports high-performance instruction execution. See also complex instruction-set computing (CISC).

relative pathname

An OSS pathname that does not begin with a slash (/) character. A relative pathname is resolved beginning with the current working directory. See also OSS pathname, absolute pathname, and current working directory.

Remote Duplicate Database Facility (RDF)

The Tandem software product that assists in disaster recovery for OLTP production databases, monitors database updates audited by the TMF subsystem on a primary system, and applies those updates to a copy of the database on a remote system.

Remote Server Call (RSC)

A Tandem software product that facilitates client/server computing, allowing personal computer(PC) or workstation applications running under Microsoft Windows software or the MS-DOS or OS/2 operating system to access Pathway server classes and Guardian processes. Transactions are transmitted from the PC or workstation application (the client) to a Pathway application running on a Tandem NonStop system (the server) by means of a supported communications protocol, such as NETBIOS, TCP/IP, or an asynchronous connection.

reply message

The part of an interprocess communication message that is formatted by a server and returned to the original requestor. The message contains the results of the server's work. See also request message.

reply translation header

A group of header fields that the Pathway translation server for the NonStop TUXEDO system inserts at the beginning of each reply message it relays from the application service to a SCREEN COBOL or Pathsend requestor. This header indicates whether the request was processed successfully and whether the reply contains data returned by the application service. See also request translation header.

request message

The part of an interprocess communication message that is formatted by a requestor and sent to a specific server. The message contains any data and instructions needed by the server to perform its processing. See also reply message.

request translation header

A group of header fields that must be included at the beginning of each request message a SCREEN COBOL or Pathsend requestor sends to the Pathway translation server for the NonStop TUXEDO system. This header specifies the NonStop TUXEDO application service for which the message is destined, the NonStop TUXEDO buffer types of the request and reply messages as seen by the service, and options that modify the invocation of the service. The translation server removes this header from the request message before sending it to the application service. See also reply translation header.

request/response server

See context-free server.

requestor

A process or program that runs in the Guardian environment on a Tandem NonStop system and requests services from a server process. For example, a SCREEN COBOL program is a requestor program that is interpreted by the terminal control process (TCP), which provides link access to Pathway server classes. Another type of requestor program makes requests through Pathsend procedure calls; such a requestor uses the ACS subsystem process for link access to server classes. A third type of requestor communicates with server processes directly by calling the Guardian WRITEREAD procedure; this kind of requestor does not use server classes. A requestor is a specific type of client. See also client, server, and requestor/server model.

requestor/server model

A model for application design that divides the tasks of data input, data manipulation, and data output between two basic types of process: requestors and servers. A requestor sends a request to a server. The server takes the requested action and then replies to the requestor. The requestor and server may reside on the same processor or on different processors. This model is used for interprocess communication in the Guardian environment. See also requestor and server.

reserved word

A word that can be used only as a keyword.

resources

The components of a computer system that work together to process transactions. Terminals, workstations, CPUs, memory, I/O controllers, disk drives, processes, files, and applications are examples of resources.

response time

The amount of time it takes to receive a response from the system after initiating a request message (for example, by pressing a function key).

retryable operation

An operation that can be interrupted and repeated an indefinite number of times without affecting the consistency of the database; for example, all read operations are retryable.

RISC

See reduced instruction-set computing (RISC).

root directory

An OSS directory associated with a process that the system uses for pathname resolution when a pathname begins with a slash (/) character. See also OSS pathname.

ROUT

Redirector (RD) processes that provide LINKMON-like functionality to Pathsend requestors. ROUT is also referred to as RD process.

NOTE: The RD is part of the ACS subsystem.

RSC

See Remote Server Call (RSC).

scalability

The ability to increase the size and processing power of an online transaction processing system by adding processors and devices to a system, systems to a network, and so on, and to do so easily and transparently without bringing systems down. Scalability is also sometimes called expandability.

SCOBOLX

The object file for the SCREEN COBOL compiler program. This name is given in a TACL command to invoke the compiler. See also SCREEN COBOL.

screen

A group of data fields that represent formatted data to be displayed on a terminal. A screen is defined by a screen description entry in the Screen Section of a SCREEN COBOL program. There are two types of screen: base screens and overlay screens. See also base screen, overlay screen, and screen description entry.

SCREEN COBOL

A procedural language developed by Tandem and based on COBOL that is used to define and control screen displays on terminals and other input/output devices. SCREEN COBOL allows programmers to write requestor programs that communicate with operator terminals and intelligent input/output devices, and that send data to server processes that manage application databases. SCREEN COBOL programs are compiled into pseudocode form by the SCREEN COBOL compiler and then interpreted by the TCP. See also terminal control process (TCP).

SCREEN COBOL Utility Program (SCUP)

A utility that provides control and manipulation of SCREEN COBOL object files.

screen description entry

A declaration of a base screen, and, optionally, an overlay screen, in the Screen Section of a SCREEN COBOL program. See also screen, base screen, and overlay screen.

screen overlay area

See overlay area.

screen program

A SCREEN COBOL requestor program. See also SCREEN COBOL.

Screen Section

A section in the Data Division of a SCREEN COBOL source program that describes the types and locations of fields in screens that can be displayed on a terminal.

screen-oriented requestor

A SCREEN COBOL requestor that sends data from working storage to the display screen of a terminal by way of screen templates defined in the Screen Section of the Data Division. Similarly, such a requestor receives data from the terminal into working storage by way of Screen Section templates. It uses ACCEPT and DISPLAY statements in the Procedure Division to interact with the display terminals. Standard SCREEN COBOL requestors are screen-oriented. See also message-oriented requestor.

SCUP

See SCREEN COBOL Utility Program (SCUP).

SEND operation

In SCREEN COBOL, an operation in which a transaction request message is sent to a server process and a reply is received back from the server process. See also server-class send operation.

server

- A process or program that provides services to a client or a requestor. Servers are designed to receive request messages from clients or requestors; perform the desired operations, such as database inquiries or updates, security verifications, numerical calculations, or data routing to other computer systems; and return reply messages to the clients or requestors. A server process is a running instance of a server program.
- A combination of hardware and software designed to provide services in response to requests received from clients across a network. For example, Tandem's Himalaya servers provide transaction processing, database access, and other services. (In the NonStop TS/MP and Pathway/TS manual set, the word "server" is generally used only when definition 1 is meant; for definition 2, "system" is usually used instead of "server.") See also client, requestor, client/server model, and requestor/server model.

server class

A group of duplicate copies of a single server process, all of which execute the same object program. Server classes are configured through the PATHMON process.

SERVER object

A definition of a server class within the configuration of a PATHMON process.

server-class send operation

The sending of a message to a Pathway server class by making a call to the Pathsend SERVERCLASS_SEND_ procedure. See also SEND operation.

service

A function performed by a server process or program on behalf of a requestor or client. A server can perform one or several services. The concept of a service is built into the design of the Novell TUXEDO system and the NonStop TUXEDO system; for these products, a service is a module of application code that carries out a service request.

simple token

In the Subsystem Programmatic Interface (SPI), a token consisting of a token code and a value that is either a single elementary field, such as an integer or a character string, or a fixed (nonextensible) structure. See also extensible structured token and token (definition 2).

single-threaded

A programming model that provides a single thread of control within a program. For example, a single-threaded server handles only one request at a time and must complete that request before accepting another. See also thread and multithreaded.

special register

A data item defined by the SCREEN COBOL compiler, rather than explicitly in the program. Each special register has a particular purpose and must be used only as defined. The SCREEN COBOL language defines a different set of special registers from those defined by the standard COBOL language.

SPI

See Subsystem Programmatic Interface (SPI).

static server

A server process that the PATHMON process creates when a START SERVER command is issued. The PATHMON process starts the number of static servers defined by the NUMSTATIC attribute for the server class. See also dynamic server.

Structured Query Language (SQL)

A relational database language used to define, manipulate, and control databases.

subsystem

In the context of the Subsystem Programmatic Interface (SPI) and the Event Management Service (EMS), a process or collection of processes that gives users access to a set of related resources or services. A subsystem typically controls a cohesive set of objects. The Pathway subsystem includes PATHMON processes, TCPs, and all other components of the PATHMON environment.

subsystem ID

In management applications, a data structure that uniquely identifies a subsystem. The subsystem ID is specified in `SPI` commands to identify the target subsystem, and in EMS event messages to identify the source of the event message. The Pathway subsystem ID applies to all components of the PATHMON environment, including PATHMON processes and TCPs.

Subsystem Programmatic Interface (SPI)

A set of Guardian procedures, message formats, and definition files that allows management applications to communicate directly with subsystem processes, such as the PATHMON process, for configuration and control of objects and for event management.

subtype 30 process

A nonprivileged process that simulates terminals and communications devices.

subvolume

A related set of files, as defined by the user, within the Guardian environment. The name of a subvolume is the third of the four parts of a file name.

swap file

A temporary file created by a process for temporary storage: for example, by the Kernel Managed Swap Facility (KMSF) on behalf of a TCP for temporary storage of terminal context.

sync ID

A value used in the Guardian environment to determine whether an I/O operation has finished. In active backup programming, a file's sync ID is used to prevent the backup process from repeating I/O operations that have already been completed by the primary process.

syncdepth

A parameter to Guardian procedure calls that sets the maximum number of operations or messages that a process is allowed to queue before action must be taken or a reply must be performed.

system name

1. The first of the four parts of a Guardian file name; also called a node name.
2. A name that identifies the Tandem system on which a PATHMON process is running. In SCREEN COBOL programs, this name is given in SEND statements.
3. A SCREEN COBOL word that identifies part of the Tandem operating environment; a name can be associated with function keys or terminal display attributes.

System/T

The transaction monitor for a NonStop TUXEDO system. See also NonStop TUXEDO application.

TACL

See Tandem Advanced Command Language (TACL).

TAL

See Transaction Application Language (TAL)

Tandem Advanced Command Language (TACL)

The user interface to the Tandem NonStop Kernel in the Guardian environment. The TACL product is both a command interpreter and a command language.

Tandem Alliance

Tandem's marketing and technical support program for third-party vendors, which encourages the development of application software for the Pathway environment. The Alliance attracts software developers, value-added resellers, and other vendors who can provide industry specific and general business applications for Tandem customers.

Tandem NonStop Kernel

The operating system for Tandem NonStop systems. The operating system does not include any application program interfaces.

Tandem NonStop Series (TNS)

Tandem computers that support the Tandem NonStop Kernel and that are based on complex instruction-set computing (CISC) technology. TNS processors implement the TNS instruction set. See also complex instruction-set computing (CISC) and Tandem NonStop Series/RISC (TNS/R).

Tandem NonStop Series/RISC (TNS/R)

Tandem computers that support the Tandem NonStop Kernel and that are based on reduced instruction-set computing (RISC) technology. TNS/R processors implement the RISC instruction set and are upwardly compatible with the TNS system-level architecture. See also reduced instruction-set computing (RISC) and Tandem NonStop Series (TNS).

task

The sequence of SCREEN COBOL program units that are executed as a result of a `PATHCOM START TERM` or `RUN PROGRAM` command or an `SPI START TERM` or `START PROG` command.

TCLPROG file

A SCREEN COBOL object library file.

TCP

See terminal control process (TCP).

TDA

See terminal data area (TDA).

TEDIT

A Tandem text editor used to create or modify a source text file. Also called PS Text Edit.

tell message

An informational message sent by PATHCOM or a management application to one or more terminals controlled by a SCREEN COBOL program, to be displayed for the terminal operators.

TELL object

A temporary object used in PATHCOM and `SPI` commands to define a tell message.

temporary TERM object

A TERM object created by the PATHMON process when a `PATHCOM RUN PROGRAM` command or an `SPI START PROG` command is issued. Temporary TERM objects are deleted by the PATHMON process when application processing is completed or when a `STOP TERM` or `ABORT TERM` command

is issued. Names of temporary TERM objects begin with a number. See also configured TERM object and TERM object.

TERM object

A definition of a task that uses a SCREEN COBOL program to control an input/output device such as a terminal or workstation, or an input/output process such as a front-end process. A TERM object can be either explicitly configured with an ADD command or created by the PATHMON process through a `PATHCOM RUN PROGRAM` or `SPI START PROG` command. TERM objects created by the latter method are called temporary TERM objects. See also configured TERM object and temporary TERM object.

terminal

An I/O device capable of sending and receiving information over communications lines.

terminal context

Data maintained by a TCP for each active terminal under its control.

terminal control process (TCP)

A process used for terminal management and transaction control, provided by Tandem as part of the Pathway/TS product. A TCP is a multithreaded process that interprets compiled SCREEN COBOL requestor programs (screen programs) in the user's application, executing the appropriate program instructions for each 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 screen programs and server processes. See also requestor and SCREEN COBOL.

terminal data area (TDA)

In SCREEN COBOL, the area that the TCP allocates for terminal context data. The `MAXTERMDATA` parameter of the `PATHCOM SET TCP` command defines the upper limit for this data area.

thaw condition

A condition in which prohibition of communication between a terminal and a server class is lifted. See also freeze condition.

thread

A task that is separately dispatched and that represents a sequential flow of control within a process (for example, a TCP).

throughput

The number of transactions a system can process in a given period, such as one second.

TMF

See Transaction Management Facility (TMF) subsystem.

TMF level recovery

Recovery of the database to a consistent state through the use of the TMF subsystem. When a failure occurs, the TMF subsystem attempts the application to back out the entire transaction, returning the contents of the database to the values it held when the transaction was started. The application can then retry the transaction.

TNS

See Tandem NonStop Series (TNS).

TNS/R

See Tandem NonStop Series/RISC (TNS/R).

token

1. An attribute control element in the CONTROLLED clause of a SCREEN COBOL program, which allows run-time control of display attributes. This token consists of an attribute identifier and an attribute value.
2. In the Subsystem Programmatic Interface (SPI), a distinguishable unit in a message. An SPI token consists of an identifier (token code or token map) and a token value. Programs place tokens in an SPI buffer by calling the SSPUT procedure and retrieve them from the buffer by using the SSGET procedure.

transaction

An operation or a series of operations that retrieves and updates information to reflect an exchange of goods or services. In the process of retrieving and updating information, a transaction transforms a database from one consistent state to another. The TMF subsystem treats a transaction as a single unit; either all of the changes made by a transaction are made permanent (the transaction is committed) or none of the changes are made permanent (the transaction is aborted).

Transaction Application Language (TAL)

A systems programming language with many features specific to stack-oriented Tandem NonStop systems. See also pTAL.

transaction backout

A TMF subsystem activity in which the effects of a partially completed transaction are canceled.

Transaction Delivery Process (TDP)

A multithreaded gateway process, part of the Remote Server Call (RSC) product, that runs on a Tandem NonStop system. The TDP can be replicated and can manage many connections and workstations at one time, as well as multiple sessions from each workstation.

transaction identifier

A unique name that the TMF subsystem assigns to a transaction.

transaction log

See audit trail.

transaction management

The ability to coordinate transaction control functions, such as beginning and ending transactions, committing or aborting transactions, and recovering transactions.

Transaction Management Facility (TMF) subsystem

The major component of the NonStop TM/MP product, which protects databases in online transaction processing environments. To furnish this service, the TMF subsystem manages database transactions, keeps track of database activity through audit trails, and provides database recovery methods. See also NonStop Transaction Manager/MP (NonStop TM/MP) and transaction.

transaction mode

The operating mode of a requestor program between the execution of a BEGINTRANSACTION procedure call or statement and the execution of the associated ENDTRANSACTION or ABORTTRANSACTION call or statement.

transaction processing

See online transaction processing (OLTP).

transaction workload

The number of transactions to be processed by an online transaction processing application.

TRANSFER delivery system

An information delivery system that allows organizations to move and manage information efficiently within a single Tandem system or across a network of systems. The TRANSFER delivery system supports communications between users, I/O devices, and processes in the Guardian environment.

TSCODE

The object code for the part of a GDSX process that is supplied by Tandem. TSCODE includes generic routines and services that support the development of a multithreaded, fault tolerant front-end process. See also USCODE and Extended General Device Support (GDSX).

tuning

The process by which a system manager allocates and balances resources for optimum system performance.

UMP

See unsolicited-message processing (UMP).

unsolicited message

A message that is sent to a SCREEN COBOL program and includes application-dependent information to be processed by the program. Although the program does not do anything to initiate the message, the message must conform to the format defined by the program. The message is sent first to the TCP. It contains a header with information that is used by the TCP and a body with information that the TCP delivers to the SCREEN COBOL program.

unsolicited-message processing (UMP)

The feature that allows terminals running SCREEN COBOL requestors to accept and reply to unsolicited messages sent to them by Guardian processes outside of the PATHMON environment.

USCODE

The object code for the part of a GDSX process that is developed by the user to provide device or access-method specifics such as control operations or data-stream translation. USCODE is bound with TSCODE to produce an operational GDSX process. See also TSCODE and Extended General Device Support (GDSX).

user conversion procedure

A procedure that lets users make their own validation checks or conversions of data passed between a SCREEN COBOL program and a terminal screen or intelligent device.

ViewPoint application

An extensible interactive application for managing operations in the Guardian environment. It provides tools for interacting with multiple Tandem subsystems, including PATHCOM, allowing a system manager to easily control an integrated Tandem system from one location.

volume

A disk drive, or a pair of disk drives that forms a mirrored disk, in the Guardian environment. The name of a volume is the second of the four parts of a file name.