HP 3000

COMMUNICATOR 3000 MPE/iX Release 5.5 (Non-Platform Software Release C.55.00)

HEWLETT PACKARD

HP Part No. 30216-90124
Printed in U.S.A.   July 1996

Volume 9, issue 1
R3628

Acknowledgements

Preface

MPE/iX, Multiprogramming Executive with Integrated POSIX, is the latest in a series of forward-compatible operating systems for the HP 3000 line of computers.

In HP documentation and in talking with HP 3000 users, you will encounter references to MPE XL, the direct predecessor of MPE/iX. MPE/iX is a superset of MPE XL. All programs written for MPE XL will run without change under MPE/iX. You can continue to use MPE XL system documentation, although it may not refer to features added to the operating system to support POSIX (for example, hierarchical directories).

Finally, you may encounter references to MPE V, which is the operating system for HP 3000s, not based on PA-RISC architecture. MPE V software can be run on the PA-RISC (Series 900) HP 3000s in what is known as *compatibility mode*.

Conventions

===>

In the Table of Contents, points to articles we recommend you read.

UPPERCASE

In a syntax statement, commands and keywords are shown in uppercase characters. The characters must be entered in the order shown; however, you can enter the characters in either uppercase or lowercase. For example:

```
COMMAND
```

can be entered as any of the following:

```
command          Command          COMMAND
```

It cannot, however, be entered as:

```
comm             com_mand         comamnd
```

italics

In a syntax statement or an example, a word in italics represents a parameter or argument that you must replace with the actual value. In the following example, you must replace *filename* with the name of the file:

COMMAND *filename*

bold italics

In a syntax statement, a word in bold italics represents a parameter that you must replace with the actual value. In the following example, you must replace ***filename*** with the name of the file:

COMMAND(***filename***)

punctuation

In a syntax statement, punctuation characters (other than brackets, braces, vertical bars, and ellipses) must be entered exactly as shown. In the following example, the parentheses and colon must be entered:

(*filename*):(*filename*)

underlining

Within an example that contains interactive dialog, user input and user responses to prompts are indicated by underlining. In the following example, yes is the user's response to the prompt:

```
Do you want to continue? >>  yes
```

{ }

In a syntax statement, braces enclose required elements. When several elements are stacked within braces, you must select one. In the following example, you must select either ON or OFF:

$$\texttt{COMMAND} \left\{ \begin{array}{l} \texttt{ON} \\ \texttt{OFF} \end{array} \right\}$$

[ ]

In a syntax statement, brackets enclose optional elements. In the following example, OPTION can be omitted:

COMMAND *filename* [OPTION]

When several elements are stacked within brackets, you can select one or none of the elements. In the following example, you can select OPTION or *parameter* or neither. The elements cannot be repeated.

$$\texttt{COMMAND}\ \textit{filename} \left[\begin{array}{l} \texttt{OPTION} \\ \textit{parameter} \end{array} \right]$$

Conventions (continued)

[...]	In a syntax statement, horizontal ellipses enclosed in brackets indicate that you can repeatedly select the element(s) that appear with the immediately preceding pair of brackets or braces. In the example below, you can select *parameter* zero or more times. Each instance of *parameter* must be preceded by a comma: [,*parameter*][...] In the example below, you only use the comma as a delimiter if *parameter* is repeated; no comma is used before the first occurrence of *parameter*: [*parameter*][,...]
| ... |	In a syntax statement, horizontal ellipses enclosed in vertical bars indicate that you can select more than one element within the immediately preceding pair of brackets or braces. However, each particular element can only be selected once. In the following example, you must select **A, AB, BA**, or **B**. The elements cannot be repeated. $\left\{\begin{matrix}A\\B\end{matrix}\right\}$ | ... |
...	In an example, horizontal or vertical ellipses indicate where portions of an example have been omitted.
Δ	In a syntax statement, the space symbol Δ shows a required blank. In the following example, *parameter* and *parameter* must be separated with a blank: (*parameter*)Δ(*parameter*)
▭	The symbol ▭ indicates a key on the keyboard. For example, RETURN represents the carriage return key or Shift represents the shift key.
CTRL *character*	CTRL *character* indicates a control character. For example, CTRL Y means that you press the control key and the character key simultaneously.

Contents

1

MPE/iX Release 5.5 (C.55.00) Overview

Communicator Summary

by The Core MPE/iX Team
Commercial Systems Division

What's New for the MPE/iX Release 5.5

This *Communicator* provides general and detailed information on the new and enhanced functionality for the MPE/iX 5.5 Release (C.55.00), as well as information on release strategy and installation prerequisites. Also included, are articles describing the functionality of the MPE/iX-Express 2 (C.50.02) and Express 3 (C.50.03) Releases, which are included in the MPE/iX 5.5 Release (C.55.00).

Pointers (==>) have been placed within the Table of Contents to help you find the articles specifically written or updated for the MPE/iX 5.5 functionality.

New Delivery Process for MPE/iX

As of MPE/iX Release 5.5, Hewlett-Packard is incorporating a new method for distributing software update materials for MPE/iX interim releases (Major, Express, PowerPatch). This software release, and all future interim releases will be easier to order. Please see the article, "New Delivery Process for MPE/iX," in this chapter for information on the new delivery/ordering process for MPE/iX interim releases.

MPE/iX Patches on HP SupportLine

MPE patches are now available on HP SupportLine via the Internet to all customers. For more details, please see the article, "MPE/iX Patches Now On the Internet," in this chapter.

Release Tapes Now in Native Mode STORE Format

Starting with Release 5.5, Hewlett-Packard will distribute HP 3000 Release tapes in Native Mode STORE format. Your installation personnel do not need to be retrained—the installation steps are unchanged. However, since Native Mode STORE generally performs better than Compatibility Mode STORE, you will probably see some significant performance improvements during your installation.

Communicator Chapters and Articles

Following are brief descriptions of the articles and chapters:

Chapter 2, System Information/Before You Install

- "Updating To and Backdating From MPE/iX 5.5" briefly describes the requirements for updating to MPE/iX 5.5 and backdating from MPE/iX 5.5.
- "Planning Your Move to MPE/iX Release 5.5" describes the requirements and procedures for updating to MPE/iX Release 5.5, including the disk space requirements.
- "Determining LDEV 1 Disk Space Using CHECKSLT " adds information on the new features of the CHECKSLT tool, which has been updated to version 1.9.
- "HP7933s and HP7935s as LDEV 1" provides information on the limitations of the HP7933s and HP7935s as LDEV 1 due to disk space requirements for MPE/iX Release 5.5.
- "AUTOINSTAL and DDS Firmware Problem" documents a problem with firmware revision 7.11 for the DDS device which may prevent an update to MPE/iX Release 5.5.
- "Recovery From OUT OF DISK SPACE During UPDATE" describes changes to UPDATE that allows UPDATE to skip any file if it cannot (for any reason) be added to LDEV 1.
- "8-MByte Memory Limitation Removed" briefly describes how the 8-MByte memory limitation was corrected so that all memory is available when loading operating system images.
- "HP System Account and Directory Naming Structure" adds information on reserved directory names and a list of these names.

Chapter 3, System Management

- "HP LaserRX/MPE Software Enhancements" describes the new functionality added to the PC LaserRX/MPE analysis software to improve the ease of use and overall functionality.
- "New JOBSECURITY Feature" describes the new parameter. PASSEXEMPT, which is used by System Managers to control password validation when the STREAM command is issued for a job file.
- "File Label - Last Modification Time" describes the file system changes for the modification date stored in the file label.
- "AIF Enhancement to AIFPROCGET Overview" provides an overview of the enhancement to AIFPROCGET, which returns connection information. See the technical article. "AIF Enhancement to AIFPROCGET Details," in Chapter 10, "Technical Articles," for more detailed information.

- "ALTSEC and HPACDPUT Enhancements" describes enhancements to the ALTSEC command and the HPACDPUT intrinsic that allow you to replace ACDs on file objects.
- "Introducing the TurboSTORE/iX 7x24 True-Online Backup Product" provides an overview of the new TurboSTORE/iX 7x24 True-Online Backup product, which lets you perform application and system backups without closing files or disrupting users. See the technical article, "STORE and TurboSTORE/iX 7x24 True-Online Backup New Functionality," in Chapter 10, "Technical Articles," for more detailed information.
- "File System Resiliency Enhancements" describes the enhancements made to the MPE/iX 5.5 Release File System.
- "Subsystem Dump Facility" describes the Subsystem Dump facility, which is part of the HP 3000 overall Software Resiliency strategy that is focusing on the avoidance of SYSTEM ABORT calls.
- "Using DAT To Examine Subsystem Dumps" describes how you use DAT to analyze subsystem dumps.
- "Introducing HP Patch/iX: A New MPE/iX Patch Management Tool" introduces the new Patch Management tool, HP Patch/iX. For a technical overview, see the technical article, "HP Patch/iX Technical Overview," in Chapter 10, "Technical Articles."
- "Introducing HP Stage/iX" describes HP Stage/iX, which is a new operating system facility for applying and managing MPE/iX patches on your system.

Chapter 4, POSIX/Open Solutions

- "Transferring HFS-Named Files With MPE Attributes (MOVER Utility)" describes a new utility, MOVER.PRVXL.TELESUP, that allows HFS files to be archived while providing expanded features.
- "CI Enhancements Overview" provides an overview of the CI enhancements that have been added in MPE/iX Release 5.5 and MPE/iX-Express 3. See the technical article, "A Detailed Look at the CI Enhancements" in Chapter 10 for details.
- "Exec Enhancement" describes the enhancement to the *exec* functions, which have been enhanced to load some non-program files, called *exec scripts*.
- "POSIX Shell Enhancements - lp and tar Commands" describes the two pieces of new POSIX shell functionality: new printing commands, `lp`, along with some associated commands; and a major enhancement to the `tar` command.
- "MPE/iX POSIX Developer's Kit Bundled with Release 5.5" provides information on the former MPE/iX POSIX Developer's

Kit, which is being bundled with the core operating system in MPE/iX Release 5.5.

Chapter 5, System Support Tools

- "Predictive Support for New Peripherals and SPUs" describes the need to update HP Predictive Support to MPE/iX 5.5 to support the new peripherals and SPUs.

Chapter 6, Application Development

- "Introducing Dependent Libraries on MPE/iX Loader" provides an overview of the new functionality, Dependent Libraries, which gives HP 3000 users an extended loading capability to include data as well as code. See the technical article, "MPE/iX Dependent Libraries Technical Overview" in Chapter 10 for details.
- "HP Link Editor/iX Enhancements Overview" provides an overview of the HP Link Editor/iX enhancements to support Dependent Libraries.
- "RPG/iX Enhancements for Version A.00.14" describes the three new enhancements added to RPG/iX.
- "VPlus Enhancements" provides an overview of the VPlus enhancements. See the article, "New Functionality in VPlus," in Chapter 10, "Technical Articles," for more details.
- "HP Symbolic Debugger/iX Enhancement" describes the enhancement to support the ability to display and modify the values of global variables whose definitions are in an executable library (XL).
- "HP Information Access Server SQL/iX Supports Newer Protocols" describes the HP Information Access Server SQL/iX enhancement that lets PC users access HP 3000 servers using Microsoft WINSOCK, or HP WSOCKETS protocols.
- "Enhancements to HP Information Access Server MPE/iX" describes the HP Information Access Server enhancements.
- "ALLBASE/4GL Developer, Release B.07.00 Enhancements" describes the new features and enhancements for ALLBASE/4GL Developer.
- "ALLBASE/SQL Enhancements" describes several major enhancements for ALLBASE/SQL version G.1.
- "IMAGE/SQL with TurboIMAGE/XL Enhancements" describes HP IMAGE/SQL, an enhanced combination of TurboIMAGE/XL and what was formerly known as ALLBASE/TurboCONNECT (ATC).
- "HP Business BASIC" describes the enhancements for HP Business BASIC/iX version A.00.15 and HP Business BASIC/V version A.02.14.

- "C/iX Library Function getenv Change in Behavior" describes the behavior change of the **getenv** function, which is a result of a defect fix in the C/iX Library.
- "Pascal/iX Run-Time Library Heap Changes" describes changes in the Pascal/iX Runtime Library (HP31502) as of version A.05.01.07.
- "Inform/V Contains Several Defect Fixes" describes the corrections made to Inform/V for the highest priority problems reported by customers.
- "Transact Contains Defect Fixes" describes the corrections made to Transact/iX and Transact/V for some of the highest priority problems.

Chapter 7, Data/Application Integration

- Announcing DCE/3000 announces the availability of the DCE/3000 product on the HP 3000 Series 900 systems.

Chapter 8, Networking/Client Server

- "Enhanced Console Switching" describes the enhancement of the console command to allow console switching over the LAN.
- "NetWare Print Enhancements" describes the enhancements added to NetWare to allow printing of MPE spoolfiles and other MPE files to LAN-connected printers.
- "Network Printer Support Now Available" provides an overview of the network printer support by the HP 3000. For more detailed information about network printing, see the "Network Printing Technical Overview" article in Chapter 10, "Technical Articles."
- "Introducing Internet Services on the HP 3000" describes the Internet Services offered on the HP 3000, which are a subset of the Internet Services available on the HP 9000, previously called the ARPA Services.
- "Introducing the Telnet/iX Server" describes the Telnet/iX Server enhancement on MPE/iX Release 5.5, which strengthens the HP 3000 as an open system by providing access between the HP 3000 and computers that support Telnet, such as UNIX-based systems and PCs.
- "DTS/TIO Dynamic Configuration and Host-Based Switching" provides an overview of the new product features of the DTS/TIO. For more detailed information, see the technical article, "DTS/TIO Dynamic Configuration and Host-Based Switching," in Chapter 10, "Technical Articles."

Chapter 9, Peripherals

- "Online System Device Configuration" describes the new MPE/iX facility, Online Device Configuration, which allows system managers and operators to configure and deconfigure new devices such as tape drives, disks and system printers while the system is online.

- "MPE/iX Host Control for Tape Drives" describes the DEVCTRL utility and the new `LOAD` feature enhancement for the MPE/iX 5.5 Release.
- "New High Performance CD-ROM Drive for MPE/iX 5.5" provides information on the HP5401 CD-ROM drive, which will be shipped as an integrated part of high-end systems 996/x00, 969/x00, and 969/x20.
- "New Optical Library for MPE/iX 5.5" describes the new HP optical library HPC1100B, which is now available for use with HP's TurboSTORE backup product.
- "New Mass Storage Systems for\MPE/iX 5.5" describes the two new mass storage systems, which use state-of-the-art disk drives, redundant power supplies (if so configured), and DDS tape drives in a fault-resilient cabinet.
- "New Disk Drives for MPE/iX 5.5" describes a series of new disk drives.
- "MPE/iX Now Supports 3.75GB of System Memory" announces the support of 3.75GB of main memory and provides related information.
- "1600 BPI Software Distribution" announces that the 1600 BPI tapes will no longer be distributed and 1600 BPI as a boot/update device will be eliminated on all HP 3000 MPE/iX products.

Chapter 10, Technical Articles

- "A Detailed Look at CI Enhancements" provides a more indepth understanding of the CI enhancements added in MPE/iX 5.5 and Express 3.
- "MPE/iX Dependent Libraries Technical Overview" provides a more indepth understanding of Dependent Libraries on MPE/iX.
- "HP Link Editor/iX Enhancements Detail" provides more detailed information on the HP Link Editor/iX enhancements to support Dependent Libraries: the new command, `ALTXL`, and the enhanced command, `BUILDXL`.
- "POSIX Libraries in XLs" provides an update of the "POSIX Libraries in XLs" article for the MPE/iX General Release 5.0 (C.50.00) *Communicator*, which corrects an error in the examples and adds a new section, "POSIX Shell Scripts."
- "STORE and TurboSTORE/iX 7x24 True-Online Backup New Functionality" provides more detailed information on the new TurboSTORE/iX 7x24 True-Online Backup product.
- "New Functionality in VPlus" provides more detailed information on the VPlus enhancements.

- "AIF Enhancement to AIFPROCGET Details" provides more detailed information on the `AIFPROCGET` enhancement.
- "HP Patch/iX Technical Overview" provides a technical overview of the new MPE/iX Patch Management tool. HP Patch/iX.
- "DTS/TIO Dynamic Configuration and Host-Based Switching Technical Overview" provides more detailed information on the new product features of the DTS/TIO.
- "Network Printing Technical Overview" provides a technical overview of network printing.
- "UPS Required for Support of New SCSI Disks" describes the need for UPS protection for the new SCSI disks introduced in the article, "New Disk Drives for MPE/iX 5.5."
- "Protective System Abort #5300 from UPS Monitor/iX" describes MPE/iX 5.5 includes a new function in the UPS Monitor/iX software that causes UPS Monitor/iX to invoke a Protective System Abort #5300 when the UPS internal battery is drained to the "low battery charge" point.

Chapter 11, Product Release History

This chapter adds product information for MPE/iX Release 5.5 and updates the termination dates in the Supported System Release Matrix table.

Chapter 12, Catalog of User Documentation

This chapter provides two types of manual listings:

- A listing at the beginning of the chapter of all new, updated, or obsoleted manuals by the time of the MPE/iX 5.5 Release.
- A listing of all manuals by manual set in alphabetical order.

New Delivery Process for MPE/iX

by Anita Doucet and Sayeh Beheshti
Software Information Distribution Organization

As of MPE/iX Release 5.5. Hewlett-Packard is incorporating a new method for distributing software update materials for MPE/iX interim releases. This software release. and all future interim releases. will be easier to order.

This article addresses commonly asked questions regarding the new delivery/ordering process for interim releases:

WHAT IS AN INTERIM RELEASE?

An interim release is a Major release (such as MPE/iX 5.5). an Express release. or a PowerPatch. These releases are available upon request by customers who have support contracts with HP.

HOW WILL AN MPE/IX SUPPORT CUSTOMER BE NOTIFIED OF A RELEASE?

With any MPE/iX interim release. you. our support customer. will directly receive a complete information and ordering packet to enable you to determine whether you want to request the new update. This packet contains:

- A cover letter.
- The MPE/iX *Communicator*.
- An order form that can be filled out and FAXed directly back to Order Processing.

  By simply filling out the order form and returning it to Hewlett-Packard via FAX or mail. your customized update release materials are shipped to you. There is no need for you to contact your Sales or Support Representative to order the release unless you desire additional information or advice.

HOW LONG DO I HAVE TO ORDER AN INTERIM RELEASE?

The length of time you have to order an interim release using this process varies depending on the type of release:

- Major release
- Express release
- PowerPatch release

An expiration date is noted on the order form enclosed in the packet. Once this expiration date has been exceeded. the normal method of contacting your Sales and/or Support Representative to order the release still applies.

WHAT IS THE COST OF THIS NEW PROCESS?

There is no additional cost to you as a Hewlett-Packard support customer as these releases are covered by your software support agreement. The new process is simply a change in Hewlett-Packard's distribution strategy.

WHY THE PROCESS CHANGE?

Hewlett-Packard is making this change because you, our support customer, requested it through your survey feedback. We are responding to your need for increased flexibility in keeping your MPE/iX environment up-to-date. The new process also allows you to receive your release materials faster than through the traditional "handtype" process as long as the order form is returned prior to the expiration date.

WHAT ABOUT PLATFORM OR CORE RELEASES?

Platform or Core releases (such as MPE/iX General Release 5.0), are automatically distributed to all customers with MPE/iX support contracts. No action on your part is required to receive these releases. In addition, Platform releases have extended support lives.

WHERE CAN I GO FOR MORE INFORMATION ABOUT THIS NEW PROCESS?

Your local Sales and Support Representatives can answer any additional questions you might have regarding the new ordering process.

MPE/iX Patches Now On the Internet

by Alice Wang
Commercial Systems Division

MPE/iX patches for MPE/iX Release 5.0 and beyond. are now available on HP SupportLine via the Internet to all customers.

Features and Benefits

The new patch access and delivery system benefits all MPE/iX customers with:

- Improved overall communication between HP and customers.
- Provision of useful and timely information for patch justification and decision making.
- Reduced system downtime for known problems.
- Reduction of the turnaround time for patch availability and delivery.
- Close to 24*7 access time.
- Unification of the MPE/iX and HP-UX patch delivery process.

Electronic access to patch information and delivery of patches provide three basic services:

1. Access to patch information in an automated. timely and accurate manner.
2. Electronic downloading of patch information and binaries.
3. Proactive notification of new patches via Email.

Access Methods to the HP SupportLine

To serve customers with different environments. HP SupportLine provides alternatives for accessing HP SupportLine. Users are encouraged to try different accessing and downloading methods to determine which are most effective for them.

Access methods include:

1. **World Wide Web Server** (WWW)

   HP SupportLine is available through the World Wide Web. The most popular browsers are the Mosaic and Netscape interfaces which are easy to use. World Wide Web access is the easiest. fastest. and most popular method of browsing for patch information and downloading patches. This is the method recommended to all users that have a choice between the Web and Email. It is more reliable. especially for large patches.

   Web accessing address: *http://us.external.hp.com*

2. **Email Server**

   The HP SupportLine Mail service is available to anyone who can send electronic mail via the Internet. Email allows HP to communicate with customers more proactively than other accessing methods. As General Release (GR) patches become available, users who have signed up for the digest are notified.

   Email accessing address: *support@us.external.hp.com*

For customers without direct Internet or Email access:

For customers who do not have Internet or Email access, an inexpensive alternative is to subscribe to an online service provider such as CompuServe, American Online, Netcom. All service providers allow customers to gain access to the World Wide Web and to send and receive electronic mail.

Electronic Digests

If you want to keep yourself up-to-date on the latest development of MPE/iX patches, you can sign up for the daily Security Bulletin and weekly mpeix_patch Bulletin. Once you have subscribed to these two bulletins, you will receive these digests on a periodic basis via electronic mail. HP SupportLine will inform you proactively about newly developed security and GR patches.

Documentation

Documents listed below contain valuable information that first-time users are encouraged to read before attempting to use HP SupportLine to download MPE/iX patches from the Internet.

- **GUIDE**: This is the HP SupportLine User's Guide for Email users only. It summarizes and gives detailed information on the SupportLine interface, its commands and functionalities.

  To request a copy, send the following to the HP SupportLine mail service:

  ```
  To: support@us.external.hp.com
  Message text:
          send guide
  ```

- **MPEGUIDE**: This document describes the components and steps required for installing an MPE/iX patch in which you have downloaded from HP SupportLine (HPSL) via the Internet.

  To request a copy, send the following to the HP SupportLine mail service:

  ```
  To: support@us.external.hp.com
  Message text:
          send mpeguide
  ```

  or

  On the "Browse or Search MPE/iX patches" HPSL WWW page, click on the `mpeguide` hyperlink.

- **README**: For every patch, there will be a README file. The file is in ASCII format. It describes the characteristics of the patch. The README file is useful for decision making when you encounter a problem. It can also help you prevent system downtime by providing information for analyzing a potential problem. The README is a document that you can read online with a Web browser or you can request HPSL Mail Service to download it for you.

  The naming convention for the README file is `RMpatchidv`.

RM:	Identifies the file as a README file.
patchid:	Indicates the unique patch ID composed of 4 alphanumeric characters.
V:	Refers to the version.

- **AUTOPATINST**: This document contains instructions to assist you in installing one or more patches needed by your MPE/iX system with the AUTOPAT installation tool.

  To request a copy, send the following to the HP SupportLine mail service:

  > `To:` *support@us.external.hp.com*
  > `Message text:`
  > *send doc autopatinst*

  or

  On the "Search Problem Solving Databases" HPSL WWW page, type AUTOPATINST in the Document ID box, and click on `Get Document`.

2

System Information/Before You Install

Updating To and Backdating From MPE/iX 5.5

by Mariann Tymn
Commercial Systems Division

Updating to MPE/iX 5.5

To update or install MPE/iX Release 5.5 (C.55.00), your system must already be on the MPE/iX 4.0 (B.40.00) or the MPE/iX 5.0 General Release (C.50.xx).

If your system is not on MPE/iX 4.0 or MPE/iX 5.0 General Release, you must first update your system to MPE/iX 4.0 before you can update to MPE/iX 5.5 (C.55.00).

If your system is on the MPE/iX 5.0 Limited Release (X.50.20), you must first update your system to MPE/iX 5.0 General Release (C.50.00) before you can update to MPE/iX 5.5.

Backdating from MPE/iX 5.5

If after installing MPE/iX 5.5 (C.55.00), you wish to backdate to an earlier release, the only supported releases that you may backdate to are MPE/iX 4.0 (B.40.00) and MPE/iX 5.0 General Release (C.50.xx). Please contact your HP Support Representative prior to performing the backdate to ensure that all options have been discussed.

Refer to the Supported System Release Matrix in Chapter 11, "Product Release History," in this *Communicator* for the expiration dates for support of MPE/iX 4.0 and MPE/iX 5.0 General Releases.

Planning Your Move to MPE/iX Release 5.5

by Joy Hansen
Commercial Systems Division

This article lists requirements for installing and/or modifying your Release 5.5 system software. This includes requirements for applying patches, adding-on subsystem (SUBSYS) purchased products, updating your system software version level, and installing your system software. If your system does not meet the requirements listed here, make the appropriate corrections prior to installing or modifying your system.

The requirements described in this article are:

- System Software Version Compatibility
- Third-Party Software Compatibility
- CD-ROM Disk Drive Compatibility
- CD-ROM with Release 4.0: Patch Requirements
- SCSI Tape Device Requirements
- LDEV 1 Disk Drive Minimum Capacity
- LDEV 1 Disk Drive Maximum Usage
- Estimating Disk Space Requirements

System Software Version Compatibility

Verify you are starting with a compatible version of the system software.

If you are running a system software version that is older than 4.0, such as 2.2 or 3.0, you must perform two updates:

1. Update to Release 4.0 using the Release 4.0 tapes and the *HP 3000 MPE/iX Installation, Update, and Add-On Manual for Release 4.0* (36123-90001).
2. Then update to version 5.5 of the system software using the 5.5 system software release media and the *HP 3000 MPE/iX System Software Maintenance Manual* (30216-90223R3628).

If you are running version 4.5 or 5.0 (limited release) system software, you must perform two updates:

1. Update to Release 5.0 (general release) using the Release 5.0 media and documentation. The media options and corresponding documentation is as follows:
   - Tapes, using the *HP 3000 MPE/iX Installation, Update, and Add-On Manual, MPE/iX Release 5.0 (General)* (36123-90001)
   - CD-ROM, using the *Using CD-ROM to Update Your HP 3000 System Software, MPE/iX Release 5.0 (General)* (B3159-90001)
2. Then update to version 5.5 system software using the 5.5 system software release media and the *HP 3000 MPE/iX System Software Maintenance Manual* (30216-90223R3628).

Contact your HP support representative if you need more information.

Third-Party Software Compatibility

Verify that any third-party software product you are running is compatible with the latest version of the operating system software. Do this *before* you modify the system.

CD-ROM Disk Drive Compatibility

To update or add-on to your system software using a CD-ROM, you must have a CD-ROM drive installed and configured. Verify that the CD-ROM drive you are using is one of the following:

- HP Series 6100 Model 600/A HP-IB (C1707A)
- HP Series 6100 Model 700/S SCSI (A1999A)
- Toshiba XM-3401TA
- Toshiba XM-4101TA

An Upgrade Kit (C2293U) for the Series 6000 peripheral package allows you to use internal SCSI drives on some HP 3000 computer systems.

If you do not have one of these compatible drives, contact your HP representative to order one.

CD-ROM with Release 4.0: Patch Requirements

If your system software version is Release 4.0 and you are using a CD-ROM to update to your system software, you need the following patches, or their supersedes, prior to updating to Release 5.5:

- MPEFX00
- MPEFX25
- MPEFX37

If you have applied any 4.0 PowerPatch, you already have these patches on your system.

You must install these patches on your 4.0 system prior to using CD-ROM media to modify your system software.

Contact your HP support representative for a copy of these patches or install them on your system using any 4.0 PowerPatch tape.

SCSI Tape Device Requirements

If your system software version is Release 4.0 and you have a SCSI-DDS tape device:

You need to verify that your SCSI-DDS devices are using a compatible version of firmware.

You **must** perform this activity **before** you update to Release 5.0 or greater. The diagnostic tools used to identify the firmware version level of the SCSI tape devices are password protected as of MPE/iX Release 5.0.

The following is a list of the SCSI-DDS devices and the compatible firmware version levels.

SCSI Tape Devices Compatible Firmware Versions

Device	Firmware Version
C152x	10.7
C15x3	10.7
C15x4	10.7

If your SCSI-DDS device is using any version level other than the one listed:

You **must** update to the compatible firmware version level before proceeding with the update. If you do not have compatible firmware for your SCSI-DDS devices and you attempt to update your system software, `AUTOINST` will fail.

Refer to the *Communicator* article "AUTOINSTAL and DDS Firmware Problem" for directions on checking and updating your DDS firmware version.

LDEV 1 Disk Drive Minimum Capacity

LDEV 1 requires a minimum capacity of 500 MBytes. Therefore, you cannot use an older HP7933 or HP7935 disk drive as LDEV 1 (the system disk).

When you update to MPE/iX Release 5.5, you may still use HP7933 and HP7935 drives elsewhere on your system, but these drives do not have a large enough capacity to serve as the system disk.

To identify your LDEV 1 device, use either the `SHOWDEV` 1 or `DSTAT ALL` command to determine the logical device numbers of configured devices.

- If the device configured as your LDEV 1 is one of the supported devices, proceed with your system modification (update, add-on, patch).
- If the device configured as your LDEV 1 is not one of the supported devices, you **must** replace your LDEV 1 device with a supported device.

Refer to the *Communicator* article "HP7933s and HP7935s as LDEV1" for additional information.

LDEV 1 Disk Drive Maximum Usage

If you have a CIO system and you are using a disk drive with over 2 Gbytes capacity as your LDEV 1, any disk space over the 2 Gbytes is not available for use. It cannot and will not be used for system or user files.

If you have an NIO system and you are using a disk drive with over 4 Gbytes capacity as LDEV 1, any disk space over the 4 Gbytes is not available for use. It cannot and will not be used for system or user files.

The predefined variable *hpcpuname* contains the name of your computer model.

```
:SHOWVAR hpcpuname
```

The following is a sample reply.

```
HPCPUNAME = Series 957
```

The following is a list of NIO/CIO systems.

NIO/CIO Systems

NIO Systems	CIO Systems (Support CIO and NIO Cards)	CIO Systems
Series 9*x*7RX, 9*x*7LX, 9*x*7SX, 9*x*8LX, 9*x*8RX, 9*x*9KS, 99*x*	Series 920, 922, 932, 948, 958	Series 925, 935, 949, 950, 955, 960, 980

Estimating Disk Space Requirements

Before you begin to modify your system software make sure that you have enough disk space to complete the modification. Modifying system software includes: updating the version level, or reinstalling your system software, adding on purchased products from the SUBSYS tape, or applying patches. There are three types of disk space requirements that are referenced during the modification process:

- The permanent (net) amount of non-contiguous disk space required by the system software after it is modified.
- The maximum (peak) amount of non-contiguous disk space required during the system software modification process.
- The amount of contiguous disk space the UPDATE tool (for example, `AUTOINST`) requires to modify the system software.

Note

The disk space values listed in this section are estimated values only. The actual amount of disk space used on your system will vary.

For additional information refer to the *Communicator* article "Determining LDEV 1 Disk Space Using CHECKSLT".

Non-Contiguous Disk Space Estimates

The table below lists the amount of non-contiguous disk space sectors required for the three operating system components (SLT, FOS, SUBSYS) and PowerPatch of the currently supported versions of the operating system software. During the modification process, some files are duplicated temporarily, sometimes older versions of files are retained temporarily. This causes the "in process" (or peak) amount of disk space usage to be greater than the "final" (or net) amount of disk space usage when the modification is complete.

You must have the peak amount of disk space available on your system during the process to successfully modify your system.

The combined System Load Tape (SLT) and Fundamental Operating System (FOS) are the minimum net requirements for any system software version level. Ensure that you have enough room on LDEV1 for all the SLT files. The FOS files do not have to go on LDEV 1.

Disk Space Sectors Required for Supported Releases

System Software Components	Disk Space Sectors for Version Level			
	4.0	5.0	5.5	
			Net	Peak
SLT only[1]	1,101,000	1,318,000	n/a	1,268,000
SLT and FOS	1,539,000	2,027,000	2,378,000	2,092,000
SLT, FOS, SUBSYS	2,905,000	3,226,000	3,637,000	3,170,000

1 Provided for reference only. An operational system requires the SLT and FOS files at a minimum.

To estimate the amount of additional permanent (net), non-contiguous disk space required to update your system software, perform the following calculation:

1. Refer to the table above and find the disk space sectors value for the system software version you are updating to.
2. Refer to the table above and find the baseline amount of disk space you are using for your current system software version.
3. Subtract the two values. This is the amount of permanent additional disk space you need to update the system software.

Disk space new version	
Disk space current version	–
Additional disk space required	=

The values listed for the SUBSYS (purchasable sub-system products) are the maximum possible. This value is the total disk space sectors required for all possible purchasable products. Typically, you will have ordered only selected SUBSYS products.

To estimate purchased product (SUBSYS) disk space:

1. You will need to estimate your disk space requirements based on the number and kind of software subsystem products purchased.

   If you are not adding-on any new purchased products, the disk space sectors currently being used by your existing products will remain the same.

   Refer to "Converting Between Disk Sectors and MBytes" below for conversion equations, if needed.

Converting Between Disk Sectors and MBytes

If your various products are listed in MBytes, particularly third-party products, estimate the disk space sectors by performing the following conversion:

n MBytes x 1,000,000 sectors/256 bytes = m sectors

Where:

n = the number of MBytes

m = the number sectors

Example 1, MBytes to sectors:

`VALIDATE` reports 20 MBytes total on a tape, so:

20 x 1,000,000/256 = 78,125 sectors

Example 2, sectors to MBytes:

A disk has a device size: 7,824,336 sectors, so:

7,824,336 x 256/1,000,000 = 2,003 MBytes

Therefore the disk is a 2000 MByte disk, (also known as 2 Gbyte disk).

For a closer estimate, use the value 1,048,576 instead of the value 1,000,000.

For a quick estimate, use the value 4,000 instead of the conversion value of 1,000,000/256.

Contiguous Disk Space Requirements

The maximum amount of contiguous disk space sectors required to complete a system modification on any system is:

120,000 sectors

Use `CHECKSLT` to estimate the amount of contiguous disk space sectors required to complete a system modification on **your** system.

Refer to the *HP 3000 MPE/iX System Software Maintenance Manual* (30216-90223R3628) for directions on using CHECKSLT.

Reserving Disk Space

The update and add-on process requires a minimum number of contiguous and non-contiguous disk space sectors. Ensure that you have enough contiguous disk space on LDEV 1 to complete your task. The total amount of non-contiguous disk space does not need to fit entirely on LDEV 1. The maximum contiguous and non-contiguous disk space requirements are listed in the table below.

Reserving Disk Space Values in Sectors

Type	4.0 to 5.5	5.0 (General) to 5.5
Contiguous	120,000	120.000
Non-contiguous	TBD	TBD

The contiguous value matches the maximum amount needed by the UPDATE tool.

The non-contiguous value is calculated as the difference between the following two values. Refer to "Non-Contiguous Disk Space Estimates" earlier in this section for the values.

- The maximum amount needed for peak installation of the SLT. FOS. full SUBSYS. and full PowerPatch.
- The maximum amount needed for permanent installation of the current version of the system software on your system.

To reserve disk space:

1. Reserve contiguous and non-contiguous disk space.

   ```
   :BUILD AXLDEV1;DISC=n,1,1;DEV=1
   :BUILD AXLSPACE;DISC=m,32,32
   ```

   Where:

   n = AXLDEV1 number. determined in "Contiguous Disk Space Estimates" or use the default of 120.000 sectors.

   m = AXLSPACE number. use the maximum value "**TBD**" (for 4.0 versions) and "**TBD**" (for 5.0 General versions).

   If a colon (:) prompt is returned. The files were built and you have enough disk space.

   If you receive a message:

   ```
   Out of disk space
   ```

You need to make more space available on your system **before** you perform an install, update, or add-on. Refer to Appendix C in the *HP 3000 MPE/iX System Software Maintenance Manual* (30216-90223R3628) for information on finding additional disk space.

2. Purge the AXLSPACE and AXLDEV1 files.

```
:PURGE AXLSPACE
:PURGE AXLDEV1
```

Determining LDEV 1 Disk Space Using CHECKSLT

by Fred Parkes
Commercial Systems Division

CHECKSLT version 1.9, a tool with a new option useful for planning your system UPDATE disk space requirements, is available with this MPE/iX release. In addition, CHECKSLT can validate any factory or Customized System Load Tape (CSLT). CHECKSLT is available on the Fundamental Operating Software (FOS STORE) release tape.

Preparing for a System Release UPDATE

The MPE/iX UPDATE process requires that you reserve some contiguous disk space on LDEV 1 prior to the UPDATE to ensure the new system files will fit on LDEV 1. The amount you need to reserve varies depending on two factors:

1. The system files that are currently on LDEV 1.
2. The files on the SLT that you intend to UPDATE to or with.

CHECKSLT calculates the required contiguous disk space by comparing the size of each file on the SLT with the size of its corresponding file on LDEV 1.

It then reports the total contiguous disk space that needs to be reserved for the UPDATE.

CHECKSLT can be invoked at any time before the UPDATE while the system is still servicing users. This way you can know in advance how much contiguous LDEV 1 disk space you need. You can retrieve the tool and catalog from the MPE/iX Release 5.5 FOS STORE tape and use it before your actual MPE/iX Release 5.5 UPDATE. (For specifics, please refer to the section "Retrieving CHECKSLT" later in this article.) During system preparation time for the UPDATE, when instructed to do so in the *HP 3000 MPE/iX System Software Maintenance Manual Release 5.5* (30216-90223R3628), reserve the amount of contiguous disk space that was reported by CHECKSLT.

Validating a CSLT Created During System Backup

During normal system operation, you can use CHECKSLT to validate any CSLT (for media and tape format errors); for example, one created during a system backup. It will catch any errors on the CSLT, and also report information for each file on the tape. You can set the level of detail CHECKSLT reports for the following:

1. Filenames only
2. Filenames and sizes
3. Filenames, sizes, label, and other detailed information

Other CHECKSLT Functions

CHECKSLT can also be used to extract any specific file from an SLT or CSLT or display the contents of an SLT or CSLT file in hexadecimal format.

Retrieving CHECKSLT

To use the new CHECKSLT option for estimating the contiguous disk space required for your MPE/iX Release 5.5 UPDATE, do the following:

Prior to the UPDATE, mount the MPE/iX Release 5.5 FOS STORE tape, put the tape drive online, and then type:

```
:HELLO MGR.TELESUP,MPEXL

:RESTORE *T;CHECKSLT,CKCAT000;SHOW
```

In CKCAT000, the 000 are zeros.

Estimating Disk Space Required by UPDATE

After CHECKSLT version 1.9 is restored onto your system, mount the MPE/iX Release 5.5 SLT, put the tape drive online, and invoke CHECKSLT, type:

```
:CHECKSLT
```

At the main menu, choose Level 7, which is described as:

```
  7 - Check the tape and display summary of tape and
disk use statistics
```

After the entire tape has been read and the file sizes are compared, the estimated amount of contiguous disk space that should be reserved on LDEV 1 is reported. Version 1.9 of CHECKSLT will compute an amount of non-contiguous disk space required on LDEV 1 if any is required. The non-contiguous disk space is in addition to the contiguous disk space required.

More Information

At the main menu of CHECKSLT choose Level 8, which is:

```
8   -  Information.
```

This level provides some details about the program and its new features.

Determining the Status of the CSLT

If you want to programmatically determine the status of the CSLT you can examine the CI variables: CHECKSLT_ERROR_FOUND, CHECKSLT_WARN_FOUND, CHECKSLT_MEDIA_WARN, CHECKSLT_DUPFILE_FOUND, and CHECKSLT_OFF_LDEV1.
This is useful when running CHECKSLT from a job. The values of the CI variables are:

0	No error or warning found.
1	Error or warning condition found.
-1	This feature was not checked during this invocation of CHECKSLT.
-2	Internal program error.

HP7933s and HP7935s as LDEV 1

by Amy Blocher and Daun Jacobsen
Commercial Systems Division

LDEV 1 requires a minimum capacity of 500 MBytes. Therefore, you cannot use the older HP7933 or HP7935 disk drives as LDEV 1 (the system disk).

When you update to MPE/iX Release 5.5, you may still use HP7933 and HP7935 drives elsewhere on your system, but these drives do not have a large enough capacity to serve as the system disk.

Refer to the *Communicator* article, "Planning Your Move to MPE/iX Release 5.5," in this chapter for additional information on LDEV 1 requirements.

To identify your LDEV 1 device, use either the `SHOWDEV 1` or the `DSTAT ALL` command to determine the logical device numbers of configured devices.

- If the device configured as your LDEV 1 is one of the supported devices, proceed with your system modification (update, add-on, patch).
- If the deviced configured as your LDEV 1 is not one of the supported devices, you **must** replace your LDEV 1 device with a supported device.

To replace your LDEV 1 device:

1. Contact your HP representative to schedule installation of the replacement disk drive.
2. Create a complete set of backup tapes (including a CSLT) containing all files residing on the system volume set.
3. Install the replacement disk drive for LDEV 1. Your Hewlett-Packard Customer Engineer performs this hardware installation.
4. Choose one of the following methods to rebuild your system and update your system software:

   Method 1:

   a. Reinstall the version of the system software that you are currently running.

      Follow the directions for a reinstallation in the installation manual for your current system software version, *HP 3000 MPE/iX Installation, Update, and Add-On Manual, MPE/iX Release (4.0 or 5.0 General)* (36123-90001).

   b. Through SYSGEN, update the I/O configuration to:

      - Reflect the new LDEV 1 Hewlett-Packard device part number.

- Reflect the new primary path of the LDEV 1 device, **if** the primary path of the hardware changed.

c. Make the update, add-on, and/or patch modifications you originally planned to do to your system.

   Follow the directions in the *HP 3000 MPE/iX System Software Maintenance Manual, Release 5.5 (C.55.00)* (30216-90223R3628).

Method 2:

a. Install (not just update) the new version of the system software.

   Follow the directions for an installation in the *HP 3000 MPE/iX System Software Maintenance Manual, Release 5.5 (C.55.00)* (30216-90223R3628).

b. Through SYSGEN, update the I/O configuration to:

   - Reflect the new LDEV 1 Hewlett-Packard device part number.
   - Reflect the new primary path of the LDEV 1 device, **if** the primary path of the hardware changed.

To determine which method to use:

- Method 1 is the safest method. It clearly defines all the processes you need to complete to properly replace your LDEV1.
- Method 2 is a short-cut method. It essentially combines the reinstallation step and the update step into one step. Do not do this unless you are completely comfortable with the installation and update processes.

5. Perform the steps in **all** the sections of Chapter 7, "Finishing the Process" in the *HP 3000 MPE/iX System Software Maintenance Manual, Release 5.5 (C.55.00)* (30216-90223R3628), excluding the section, "Permanently Applying a Staging Area."

   These activities:

   - Ensure that your configuration file is cross validated with your SYSGEN information.
   - Restore the user files which had previously resided on the system volume set.

Note Issue all RESTORE commands with the KEEP option to prevent a mismatch of two different operating system versions.

AUTOINSTAL and DDS Firmware Problem

by Lori Orvek and Larry Nichoalds
Commercial Systems Division

Certain firmware revisions for the HPC1503B/HPC1520B DDS device have a defect which will prevent some files from being restored from `CMSTORE` format tapes. specifically the FOS and SUBSYS tapes. As a result of this defect, an AUTOINSTAL from this device may fail and the spoolfile output of the restore will show an error message similar to `UNEXPECTED END OF FILE MARKER FOUND (S/R 9060)`.

If your system software version is Release 4.0 and you have a SCSI-DDS tape device:

> You need to verify that your SCSI-DDS devices are using a compatible version of firmware.
>
> You **must** perform this activity **before** you update to Release 5.0 or greater. The diagnostic tools used to identify the firmware version level of the SCSI tape devices are password protected as of MPE/iX Release 5.0.

Refer to the *Communicator* article. "Planning Your Move to MPE/iX Release 5.5" in this chapter, for additional information on updating from various versions.

The following is a list of the SCSI-DDS devices and the compatible firmware version levels.

SCSI Tape Devices Compatible Firmware Versions

Device	Firmware Version
C152x	10.7
C15x3	10.7
C15x4	10.7

If your SCSI-DDS device is using any version level other than the one listed:

> You must update to the compatible firmware version level before proceeding with the update. If you do not have compatible firmware for your SCSI-DDS devices and you attempt to update your system software, `AUTOINST` will fail.

Note If you are modifying a Remote system, check the SCSI-DDS devices on each remote system in addition to the local system.

To identify the firmware version of each of your DDS devices, for each device, perform the following:

Caution This procedure must be done **before** you update to 5.0 or greater.

1. Start the system diagnostic tool.

```
:sysdiag
dui> scsidds;ldev=dds_ldev;section=50
```

   Where *dds_ldev* is the logical device number assigned to the SCSI-DDS device. A common *dds_ldev* value is 7.

2. Identify the firmware revision.

```
scsidds> rev
```

   SCSIDDS displays the firmware revision. The following is a sample output.

```
Firmware Rev = 10.7
Servo Rev = 2.2
```

3. Exit the SCSIDDS and SYSDIAG utilities.

```
scsidds> exit
dui> exit
```

   If the firmware revision displayed does not match the required firmware version level, *do not proceed with the system modification until after the device firmware has been corrected*. To change the device firmware, contact your HP representative.

Recovery From "OUT OF DISK SPACE" During UPDATE

by Fred Parkes
Commercial Systems Division

This MPE/iX release includes changes to UPDATE that allows UPDATE to skip any file if it cannot (for any reason) be added to LDEV 1. This means that it is now possible to recover from "OUT OF DISK SPACE" during UPDATE. You will see error and status messages on the console at the point the file is skipped. There is also an error message at the end of UPDATE that provides some information and direction.

What You See

At the end of UPDATE, when the system normally boots, the system will stop. The operator must boot the machine manually. Messages will stay on the console. The console will have a form of the following message on it.

```
               ERROR.

   This UPDATE did not restore all files from the tape.      (UPDERR 1001)
   There were        # files not restored because of out of disk space.
   There were        # files not restored for other than disk space reasons.

   There is more information about the error conditions in previous messages.
   START the system, create more free disk space and run this UPDATE again.
   Correct the problems and run this UPDATE again.
   Correct the problems, START the system, create more free disk space and
   run this UPDATE again.
   END of LOAD(update).
```

What You Should Do

If files have been skipped during an update, follow these steps to correct the problem:

1. Determine the reason the files were skipped.
2. Start the system.
3. Correct the problem.
4. Shut the system down.
5. Do the update again.
6. After all files on the SLT have been restored to disk, continue with the task at hand.

Files Skipped Because Out of Disk Space

If there were one or more files skipped because out of disk space, look at the messages on the console to determine which files were skipped. If the files ISL.MPEXL.SYS, MMSAVE.MPEXL.SYS, and START.MPEXL.SYS have been restored to disk:

1. Boot the system.
2. Free some disk space on LDEV 1.
3. Shut the system down.
4. Redo the update from the same tape.

The files ISL.MPEXL.SYS and MMSAVE.MPEXL.SYS are small so there should not be a problem restoring them.

The file START.MPEXL.SYS is large but has been given special consideration. Instead of just skipping it, the file DUMPAREA.MPEXL.SYS is purged (by UPDATE) to make room for START.MPEXL.SYS. Under any reasonable conditions the file START.MPEXL.SYS is written to disk and there is at most only one file skipped.

Reasonable conditions are: the file DUMPAREA.MPEXL.SYS can be purged to make room for START.MPEXL.SYS and there are no other problems. If the file DUMPAREA.MPEXL.SYS is purged, it will be replaced by UPDATE when enough disk space is available.

Files Skipped Because of Other Reasons

If there were files skipped for other than disk space reasons, determine the cause. There will be error messages and status on the console. When a call to HPFOPEN returns an error status, that status is always displayed on the console. This is the key to the cause of the problem. These error statuses are the same statuses that any caller to HPFOPEN can receive. Whatever the problem, it must be corrected.

1. Correct the problems that have been found.
2. If there are still problems to correct, start the system and correct them.
3. If you started the system, shut the system down.
4. Update again from the same tape.

Missing Messages

If error messages have scrolled off the screen and you need the information, try the update again. The same conditions should exist and the same messages should be generated.

Mix and Match

If a system experiences a problem and files have been skipped. the system on disk is complete except for the skipped file or files. It is safe to start the operating system. The system will not be a mixed system on disk. The process of replacing files consists of first purging the old file and then creating the new file. Any errors that occur are found between the point where the old file is purged and the new file is created.

If This Does Not Work

If the steps above do not resolve the problem. contact your support representative. Trained Hewlett-Packard response center engineers can recover some systems.

Things to Remember

When to Continue

If you have experienced problems during UPDATE. you can continue when the following conditions are met:

- UPDATE has completed successfully.
- All files from the SLT have been restored to disk.
- The system is up and running.
- There is some free space on LDEV 1.

Reserving Contiguous Disk Space for UPDATE

Before starting UPDATE you are asked to reserve an amount of contiguous disk space in the file AXLDEV 1.PUB.SYS. This amount is at least 60.000 sectors but may be larger. The prudent way to reserve this space is in one contiguous block. Reserving this space in more than one block has the affect of causing fragmentation. It introduces a risk that Secondary Storage Manager (SSM) will allocate space for files on LDEV 1 in a way that causes an "out of contiguous disk space" condition during UPDATE. Always try to reserve this contiguous disk space for UPDATE in one block.

8-MByte Memory Limitation Removed

by Fred Parkes
Commercial Systems Division

With MPE/iX Release 5.0, the 8-MByte memory limitation has been removed.

In the MPE/iX Release 4.0 several low-end system models had a hardware limitation due to the 8-MByte memory controller cards in them. For more information, refer to the MPE/iX Release 4.0 PowerPatch B.40.08 Addendum.

This problem has been corrected in Release 5.0 (and later) by making all memory available when loading operating system images.

In MPE/iX General Release 5.0 a new module, TAPEIPL, was added for tape boot-up. If you are booting from tape with MPE/iX Release 5.5, you will see this new module as the first module from the tape, before ISL.

No action is required - this is for your information only.

HP System Account and Directory Naming Structure

by The Release Delivery Team
Commercial Systems Division

Since the MPE operating system originated. Hewlett-Packard has developed and maintained a number of system accounts that are considered reserved. With the addition of POSIX. this idea is extended to include a number of Hierarchical File System (HFS) directories as well. Each time you perform a system installation or update. the new information added to your system is placed in these reserved structures.

Hewlett-Packard recommends that you do not use reserved names to avoid overwriting your user accounts and directories. Rename all non-Hewlett-Packard accounts and directories that use any of the reserved names.

System Accounts

The following accounts are currently in use:

CONV
CLL
HPLANMGR
HPNCS
HPOFFICE
HPOPTMGT
HPPL85
HPPL87
HPPL89
HPSKTS
HPSPOOL
HPX11
ITF3000
RJE
SNADS
SUPPORT
SYS
SYSMGR
TELESUP

System Directories

The following directories are currently in use:

```
/bin                              /usr/lib/terminfo/m
/etc                              /usr/lib/terminfo/n
/hpshell-examples                 /usr/lib/terminfo/o
/hpshell-examples/lexyacc         /usr/lib/terminfo/p
/lib                              /usr/lib/terminfo/q
/tmp                              /usr/lib/terminfo/r
/usr                              /usr/lib/terminfo/s
/usr/curses                       /usr/lib/terminfo/t
/usr/include                      /usr/lib/terminfo/u
/usr/include/sys                  /usr/lib/terminfo/v
/usr/lib                          /usr/lib/terminfo/w
/usr/lib/curses                   /usr/lib/terminfo/x
/usr/lib/tabset                   /usr/lib/terminfo/y
/usr/lib/terminfo                 /usr/lib/terminfo/z
/usr/lib/terminfo/a               /usr/lib/terminfo/0
/usr/lib/terminfo/b               /usr/lib/terminfo/1
/usr/lib/terminfo/c               /usr/lib/terminfo/2
/usr/lib/terminfo/d               /usr/lib/terminfo/3
/usr/lib/terminfo/e               /usr/lib/terminfo/4
/usr/lib/terminfo/f               /usr/lib/terminfo/5
/usr/lib/terminfo/g               /usr/lib/terminfo/6
/usr/lib/terminfo/h               /usr/lib/terminfo/7
/usr/lib/terminfo/i               /usr/lib/terminfo/8
/usr/lib/terminfo/j               /usr/lib/terminfo/9
/usr/lib/terminfo/k               /usr/mail
/usr/lib/terminfo/l               /usr/man
```

3

System Management

HP LaserRX/MPE Software Enhancements

by Gary Robillard & Adrian Peterson
Worldwide Response Center Organization

HP LaserRX/MPE is a performance management software product for HP 3000 MPE V and MPE/iX systems. HP LaserRX/MPE has two major components:

- Data collection and management software that runs on the HP 3000 system
- Data analysis software that runs on an IBM compatible personal computer (PC)

There has been major new functionality added to the PC analysis software (known as the C.00.00 Release) to improve the ease of use and overall functionality.

Note

The HP LaserRX/MPE software (Product Number HP50700C) for HP 3000 systems has been included beginning with the MPE/iX-Express 2 based on General Release 5.0 SUBSYS tape.

Customers with support contracts for LaserRX should have received the update to the PC analysis software. To receive the host component only, order the HP50700C product without specifying any licensing options through your HP representative.

Overview of PC-Based Analysis Software Changes

The following changes make the LaserRX/MPE analysis software running on a PC easier to use:

New Features

- A new toolbar allows quick access to commonly used features that can be used for each graph independently.
- Graph scaling can be toggled between logarithmic and linear using a push button.
- Graph lines can be changed from solid to dashed by using a push button.
- Graph backgrounds can be toggled between black and white using a push button.

- TCP/IP stream socket connections are supported (Win Sockets and HP Sockets) to MPE/iX systems.
- Printer setup is now available from the File menu.
- Help supports keyword searching, browsing, and Microsoft Windows 3.1 context-sensitive Help.

Changed Features

- The PC analysis software is now a Microsoft Windows 3.1 application and requires Microsoft Windows 3.1 to run.
- Multiple Document Interface feature allows displaying multiple graphs at once.
- Graphs can be iconized (minimized) and restored.
- Native Language Support (NLS) has been improved.
- The status bar indicates the severity of messages through color.
- Windows REAL mode is no longer supported.

Changes to the LaserRX/MPE HP 3000 Software

The LaserRX/MPE HP 3000 software has had three modifications for the C.00.00 Release:

- The Global Transaction Rate Graph has been changed to prevent overflow on fast MPE/iX system processors (such as 98X, 99X).
- The EXTRACT program now defaults to a record limit of 65,535 for RXLOG files.
- A server has been added to allow the PC to connect to the HP 3000 using TCP/IP stream sockets connections.

Note The logfile formats have not been changed.

Global Transaction Rate Graph Change

As part of the C.00.00 Release of LaserRX/MPE, the Global Transaction rate graph has been changed from transactions X 1,000 to transactions X 100,000. This was done because the global transaction metric was overflowing on some of the faster MPE/iX system processors (such as 98X, 99X, etc).

Prior to the C.00.00 Release of LaserRX/MPE, the PC/HOST communication software was limiting the global transaction rate to 32,767 transactions per minute (1,966,020 transactions per hour). With the change on both the host and the PC side, LaserRX can display global transaction rates of up to 3,276,700 transactions per minute (196,602,000 transactions per hour).

The version of the datacomm library (XL.SCOPE.SYS) was changed from B.00.03.11 to B.00.03.12 to support this change.

The new LaserRX PC analysis software detects the version of the HP 3000 datacomm library that is being used. If the new PC analysis software (the C.00.00 version) is being used with the older HP 3000 host software, the global transaction rate graph remains as transactions x 1,000 instead of transactions x 100,000.

EXTRACT Program Change

The EXTRACT.SCOPE.SYS program has been modified so that when a new RXLOG file is created, it has a maximum of 65,535 records, instead of the old default maximum of 32,767. This prevents you from having to issue a file equation for large extractions of data. The `EXTRACT`, `WEEKLY`, `MONTHLY`, and `YEARLY` commands all use the new limit of 65,535 records when creating an RXLOG file.

TCP/IP Stream Sockets Support Added

The current LaserRX/MPE product communicates from the PC to the host HP 3000 using HP Cooperative Services. This allows the PC to communicate either over an RS232 serial connection or via a LAN connection using NetIPC.

The C.00.00 Release of LaserRX/MPE allows the use of TCP/IP stream sockets when connecting the PC to the HP 3000 using a LAN connection. The TCP/IP stream sockets protocol is available for MPE/iX but is not available for MPE V.

If you currently connect via LAN to an MPE/iX system using NetIPC then you can connect to the MPE/iX HP 3000 using TCP/IP stream sockets. To allow the use of TCP/IP stream sockets, a new version of the LaserRX/MPE HP 3000 software needs to be installed. The new version of the LaserRX/MPE HP 3000 software is only required with the C.00.00 version of the PC analysis software in order to use TCP/IP Stream Sockets connections from the PC to the HP 3000.

The PC that runs the LaserRX/MPE PC analysis software requires either WSOCKETS.DLL or WINSOCK.DLL, and must be MICROSOFT Version 3 compatible to run TCP/IP stream sockets (use WHAT.EXE for version verification).

To install the new version of the LaserRX/MPE HP 3000 software, you must install MPE/iX-Express 2 based on General Release 5.0 (or later). Call your HP Representative to order the new version of LaserRX/MP software.

If you are on a version of MPE/iX prior to General Release 5.0, you will need to contact your HP Support Representative.

New JOBSECURITY Feature

by Gail Duro
Commercial Systems Division

Overview

The JOBSECURITY command now includes a new parameter. PASSEXEMPT, in this MPE/iX 5.5 Release. PASSEXEMPT is used by users with System Manager (SM) capability to control password validation when the STREAM command is issued for a job file.

Note

Password validation for the STREAM command is also controlled by the HP Security Monitor, which is a separately purchasable product. If you already have the HP Security Monitor, use that to control password validation. Otherwise, you can use the JOBSECURITY ;PASSEXEMPT command, which is available on MPE/iX 5.5. Refer to the "JOBSECURITY Interaction with the HP Security Monitor" section later in this article for a comparison of the HP Security Monitor and the JOBSECURITY ;PASSEXEMPT command.

PASSEXEMPT can give users the ability to stream jobs without requiring the logon passwords. Normally, when the STREAM command is issued, users are interactively prompted for passwords or are required to embed them within the job file.

Intended Audience

This article is intended for System Managers (SM), Account Managers (AM), and general users who want to stream jobs without specifying passwords. This article provides a general overview of the enhanced JOBSECURITY command and the new PASSEXEMPT parameter.

Features and Benefits

When the PASSEXEMPT parameter is enabled, a set of users are exempted from the password requirement and can stream jobs without having to specify passwords. Passwords are still required for logging on to a session interactively.

Features

The PASSEXEMPT parameter provides the following features:

- The password exemption feature for streaming job files can be granted to any user with SM, AM or a matching logon identity.
  - SM users can stream all jobs.
  - AM users can stream jobs that logon to their account.
  - A user whose logon identity matches the job's logon identity can stream a job without specifying passwords.
- Additional users can be authorized to stream jobs without specifying passwords. When the stream file's owner/creator is the same as the job logon identity and the user has execute access

to the file, they are allowed to stream the job without specifying passwords. The file owner/creator can create an access control definition (ACD) to select which users they want to grant execute access.

Benefits

The benefit to System Managers is better password management. By using PASSEXEMPT, they can limit password access while still providing the ability to stream jobs.

JOBSECURITY Operation

The JOBSECURITY command may be issued from a session, job, program, or in BREAK. Pressing the BREAK key has no effect on this command. You may execute JOBSECURITY only from the console unless distributed to users with the ALLOW command.

JOBSECURITY controls the use of the ABORTJOB, ALTJOB, BREAKJOB, and RESUMEJOB commands with the HIGH or LOW parameter, described in the "Parameter Definitions" section later on.

JOBSECURITY Interaction with the HP Security Monitor

PASSEXEMPT and the HP Security Monitor perform the same password verification functions. Normally, you would use one or the other, but you can use both.

Following is a comparison of the PASSEXEMPT options and the equivalent HP Security Monitor features (see the "Parameter Definitions" section below for descriptions of these options):

PASSEXEMPT Option	**HP Security Monitor Feature**
USER	Stream Privilege
XACCESS	Stream Authorize

If you do have the HP Security Monitor on your system, and you use the JOBSECURITY command, the output of JOBSECURITY could be impacted by the HP Security Monitor. This is because the HP Security Monitor manages its settings in its own configuration file. When you invoke the JOBSECURITY command, it checks to see if the HP Security Monitor file exists. If the HP Security Monitor file does exist, JOBSECURITY combines the settings to produce the command output, which may not be the expected output.

For example, if you set PASSEXEMPT=XACCESS, and the Stream Privilege feature of the HP Security Monitor was also set, then JOBSECURITY combines these settings as if MAX were set (which combines the USER and XACCESS options).

Note

If you have the HP Security Monitor, we recommend that you set all the settings either with the HP Security Monitor, or with the `JOBSECURITY ;PASSEXEMPT` command, to be sure of the command output.

When the `PASSEXEMPT` parameter is issued and the interaction with the HP Security Monitor produces a different result, a warning that the HP Security Monitor is installed is issued. The resulting command output is also displayed with the warning. This is illustrated in the last example in the "Examples" section.

User Interface

The `JOBSECURITY` command designates what level of user may request resource and control the execution of jobs. The following information describes the parameters supported by `JOBSECURITY`:

Syntax

```
                                    ┌                 ⎧ NONE    ⎫ ┐
                         ┌HIGH┐     │                 ⎪ USER    ⎪ │
         JOBSECURITY     │    │     │ ;PASSEXEMPT=   ⎨         ⎬ │
                         └LOW ┘     │                 ⎪ XACCESS ⎪ │
                                    └                 ⎩ MAX     ⎭ ┘
```

Parameter Definitions

`HIGH` Permits only the operator logged on at the console to use job control commands. (Optional)

`LOW` Allows any user to issue job control commands for their own jobs. (Optional)

The job's username and account must match that of the user. Account Managers do not need a matching username, so they may control the execution of any job in their account.

`PASSEXEMPT` Controls password validation when the `STREAM` command is issued for a job file. SM capability is required to specify `PASSEXEMPT`. (Optional)

`NONE` Requires that the requested passwords be specified to stream a job. If the `PASSEXEMPT` parameter has never been used before and the HP Security Monitor is not installed, the initial state is `NONE`. (Default)

When the system is rebooted with `START NORECOVERY`, the `PASSEXEMPT` parameter is initialized to `NONE`.

When the system is rebooted with a `START RECOVERY`, the last `PASSEXEMPT` state is preserved.

USER	Grants password exemption to users with SM, AM, or matching logon identity. ■ The System Manager can stream all jobs. ■ The Account Manager can stream all jobs that log on to their account, provided they otherwise have access to those jobs. ■ A user can stream all jobs where their logon identity matches the job's logon identity and they have access to those jobs. The USER option is equivalent to the Stream Privilege feature in the HP Security Monitor.
XACCESS	Grants a user to stream a job without specifying a password if their logon identity matches the job's logon identity, and they have execute access to the stream file. The file owner/creator can set an access control definition(ACD) on the file to grant execute access to the specific set of users. The XACCESS option is similar to the HP Security Monitor's Stream Authorize feature. However, the Stream Authorize feature can only be enabled if the Stream Privilege feature is already enabled. The JOBSECURITY command, however, allows the options to be set independently.
MAX	Specifies both USER and XACCESS. Otherwise, these options are mutually exclusive.

If you do not specify any options, the current job security status is displayed.

Examples

The following examples show the enhanced JOBSECURITY command and changes to the output.

```
: JOBSECURITY
JOBSECURITY IS HIGH. PASSEXEMPT IS NONE.

: JOBSECURITY LOW
: JOBSECURITY
JOBSECURITY IS LOW. PASSEXEMPT IS NONE.

: JOBSECURITY ;PASSEXEMPT=USER
: JOBSECURITY
JOBSECURITY IS LOW. PASSEXEMPT IS USER.
```

If USER is set and XACCESS is specified, the result is **XACCESS**.

```
: JOBSECURITY
JOBSECURITY IS LOW. PASSEXEMPT IS USER.

: JOBSECURITY ;PASSEXEMPT=XACCESS
: JOBSECURITY
JOBSECURITY IS LOW. PASSEXEMPT IS XACCESS.
```

If the HP Security Monitor is installed with both Stream Privilege and Stream Authorize turned on, the JOBSECURITY command displays an output warning when the OR operation produces a different result.

```
    :JOBSECURITY ;PASSEXEMPT=USER

Security Monitor is installed. Passexempt is MAX. (CIWARN 3128)
```

File Label - Last Modification Time

by Pat Alvarez
Commercial Systems Division

This MPE/iX 5.5 Release includes file system changes for the modification date stored in the file label. In the past, the modification date was updated with the current date when a disk file was opened for write access. This was true even when no actual writes occurred.

Beginning with this MPE/iX 5.5 Release, the modification date in the file label is only updated when a write (update, append) is actually performed on the file. With this modification, the handling of timestamps for MPE/iX files is similar to the way they are handled in a UNIX® environment.

The modification date for mapped files and compatibility mode files are not affected by this change. The modification date in the file label is always updated when these files are opened for write access.

AIF Enhancement to AIFPROCGET Overview

by John Berry
Commercial Systems Division

The Architected Interface Facility (AIF): Operating System product (HP36374A), version A.08.00, contains an enhancement to **AIFPROCGET** in MPE/iX Release 5.5. A new item, *2149*, was added to **AIFPROCGET** to return connection information. This enhancement makes it easier for the system manager and/or operator to monitor who is connected to an HP 3000.

When **AIFPROCGET** is called with item number *2149*, it retrieves information on sockets owned by a given process (pin or pid). This item can be used to request information on all or just incoming socket connections belonging to a process. The maximum number of sockets that can be opened per process is currently 1,024.

Important Details Please Read

Socket information is available for Internet/ARPA protocol stack users only. Information for non-internet protocol stacks such as SNA, NetWare and/or Appletalk is not available through this item. Also, information for regular DTC terminals, DTC/TAC (Telnet Access Card) and TEB (Telnet Express Box) is not available.

For more details, see the technical article, "AIF Enhancement to AIFPROCGET Details," in Chapter 10, "Technical Articles."

ALTSEC and HPACDPUT Enhancements

by Michael Paivinen
Commercial Systems Division

Overview

Starting with MPE/iX Release 4.5, certain file objects are required to have Access Control Definitions (ACDs) to describe their security. These file objects include:

- Hierarchical directories
- File objects in hierarchical directories
- Files in MPE groups where the file's group ID (GID) does not match the GID of the MPE account

If a file's ACD is required, you cannot remove the ACD with the command `ALTSEC` *file* `;DELACD` (or the `HPACDPUT` intrinsic). Since the ACD cannot be removed, you cannot replace the file's ACD by doing `ALTSEC` *file* `;DELACD` followed by `ALTSEC` *file* `;NEWACD=(`*acdpairs*`)`.

To solve this problem, the keyword `REPACD` was added to the `ALTSEC` command in MPE/iX Release 4.5. This keyword replaces an entire existing ACD for a file object, or creates a new ACD if none exists.

The remainder of this article describes additional enhancements to the `ALTSEC` command and the `HPACDPUT` intrinsic that allow you to replace ACDs on file objects.

4.5 HPACDPUT Enhancements

In MPE/iX Release 4.5, functions 34 through 36 were added to the `HPACDPUT` intrinsic to allow a program to replace the ACD on an existing file object. (The file specified with function 34 is an ACD indirect file.)

HPACDPUT Itemnum2/Item2 Values

Itemnum	Mnemonic	Item Description
34	BA	Replace an existing ACD, or create a new ACD if none exists. The *item2* value specified must be an MPE- or HFS-name syntax filename. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash and the null character terminator. MPE filenames < 35 characters must be terminated with a null character or a carriage return.
35	BA	Replace an existing ACD, or create a new ACD if none exists. The *item2* value specified must be a 1- to 279-character byte array.
36	I	Copy an ACD from a previously opened file number, replacing an existing ACD, if required. The *item2* value specified must be a 16-bit integer containing the *filenum* of the file with the ACD to be copied.

5.5 ALTSEC Enhancement

Beginning with MPE/iX Release 5.5, the `REPACD` keyword has been enhanced to allow you to copy an ACD from one object to another object, replacing an existing ACD, if required. The new syntax for using `REPACD` is:

```
ALTSEC targetfile

 [          ( (acdpair[;acdpair][; ... ]) ) ]
 [ ;REPACD= ( ^indirectfile               ) ]
 [          ( sourcefile                  ) ]
```

All filenames can be specified using MPE-escape syntax.

- When using `acdpairs`, the ACD is provided on the command line.
- When using *^indirectfile*, the indirect file contains the ACD.
- When using a *sourcefile*, the ACD is copied from *sourcefile* to *targetfile*.

The first two methods are the same as on MPE/iX Release 4.5.

5.5 HPACDPUT Enhancement

Beginning with MPE/iX Release 5.5, function 37 has been added to the HPACDPUT intrinsic. This function works the same as function 36 except that the source file is specified using an MPE-escape syntax filename instead of a file number.

HPACDPUT Itemnum2/Item2 Values

Itemnum	Mnemonic	Item Description
37	BA	Copy an ACD from a specified file, replacing an existing ACD, if required. The *item2* value specified must be specified using MPE- or HFS-name syntax. HFS pathnames must begin with a dot (.) or slash (/) character and cannot exceed 1024 characters, including the dot or slash and the null character terminator. MPE filenames < 35 characters must be terminated with a null character or a carriage return.

Introducing the TurboSTORE/iX 7x24 True-Online Backup Product

by Jim Nissen
Commercial Systems Division

Overview

The TurboSTORE/iX 7x24 True-Online Backup product is now available in the MPE/iX 5.5 Release. TurboSTORE/iX 7x24 True-Online Backup enables you to perform application and system backups without closing files or disrupting users.

Important Details Please Read

The STORE/iX and TurboSTORE/iX product structure has changed from the MPE/iX 5.0 Release. See the "Product Structure Change" section later in this article for details.

TurboSTORE/iX 7x24 True-Online Backup (B5152AA) can perform two types of online backups:

- Online backup (previously part of the TurboSTORE/iX II with online backup product (36398A)), which requires all files be stored or closed before performing a backup.
- 7x24 true-online backup, which enables customers to perform both application and system backups *without* closing files or disrupting users.

  The 7x24 true-online backup does not require users to shutdown their applications. However, TurboIMAGE and ALLBASE/SQL databases must be quiesced (no new transactions can start until the open transactions are complete) for a short period to guarantee logical database consistency. TurboSTORE/iX 7x24 True-Online Backup automatically quiesces TurboIMAGE and ALLBASE/SQL databases.

This article focuses on the new 7x24 true-online backup method.

The following functionality and enhancements have been added as of MPE/iX Release 5.5 and are included in the TurboSTORE/iX 7x24 True-Online Backup product:

- New Functionality:
  - New TurboSTORE/iX 7x24 True-Online Backup options: `ONLINE` (with `START`, `END`, `ASK` parameters), and `LOGVOLSET`
  - New STORE options: `PARTIALDB`, `FULLDB`, `STOREDIRECTORY`, and `NOSTOREDIRECTORY`, and `STATISTICS`
  - Storing to disk files
  - Restoring from disk files
- Enhancements:
  - Physical consistency

- Shadow logging

New Functionality

Following are brief descriptions of the functionality added to support the TurboSTORE/iX 7x24 True-Online Backup product. For more details, refer to the technical article, "STORE and TurboSTORE/iX 7x24 True-Online Backup New Functionality" in Chapter 10, "Technical Articles," in the MPE/iX 5.5 *Communicator*.

For more in depth information, see the *STORE and TurboSTORE/iX Products Manual* (B5151-90001), which is now shipped with each release of the Fundamental Operating System (FOS) as part of the System Management Core Manual Set (36367A), and with each purchased TurboSTORE/iX product.

New TurboSTORE/iX 7x24 True-Online Backup Options

`ONLINE`	With the `START` or `END` parameters, performs backups without closing files.	
	`START`	Places the *sync point* (a reference point in time) at the beginning of a backup.
	`END`	Places the sync point at the end of the backup.

Note

If no parameters are specified for `ONLINE`, the previous online backup is used, which requires that all files be closed for write access at that time.

	`ASK`	The `ASK` parameter with `ONLINE` lets you synchronize any other file types (e.g., Compatibility Mode (CM) file types such as Circular (CIR) files and Relative I/O (RIO) files) and/or third-party databases that are not automatically synchronized. `ASK` makes STORE wait for you to finish these operations.
`LOGVOLSET`	Lets you specify which volume set to use for log files.	

New STORE Options

`PARTIALDB`	Lets you back up a part of a TurboIMAGE database or a part of an ALLBASE/SQL DBEnvironment.
`FULLDB`	In a non 7x24 true-online backup environment (i.e., `ONLINE=START` or `ONLINE=END` is not specified), lets you backup the entire database by just specifying the root filename only in the fileset.

STOREDIRECTORY	Default option when a 7x24 true-online backup is created with the sync point at the *end* of the backup. Puts a copy of a backup's STORE label and STORE directory into a disk file.
NOSTOREDIRECTORY	Default option when a 7x24 true-online backup is created with the sync point at the *beginning* of the backup. Does not allow a directory file to be created.
STATISTICS	Provides extra data about a backup.

Storing To Disk Files

TurboSTORE/iX 7x24 True-Online Backup allows you to store the data to disk.

Restoring From Disk Files

TurboSTORE/iX 7x24 True-Online Backup allows you to restore data that was stored to one disk file, multiple disk files, or parallel disk files.

Enhancements

Physical Consistency

TurboSTORE/iX 7x24 True-Online Backup ensures physical consistency for system files, some flat files, and any other files not associated with a database.

Shadow Log Files

TurboSTORE/iX 7x24 True-Online Backup now provides the capability of specifying where *shadow log* files reside (these files contain the *before* images of changes to files being backed up).

Product Structure Change

The product structure of the STORE and TurboSTORE/iX products has changed for MPE/iX Release 5.5. Some of the features have been placed or kept in the STORE/iX program in the MPE/iX Fundamental Operating System (FOS), and the rest have been consolidated into the following two products:

- TurboSTORE/iX II (new version)
- TurboSTORE/iX 7x24 True-Online Backup (new product)

The following table shows how the previous TurboSTORE/iX product structure for MPE/iX Release 5.0 has been incorporated into the current product structure for MPE/iX Release 5.5.

Product Structure Change

5.0 Product Structure			5.5 Product Structure	
Product #	**Product Description**		**Product #**	**Product Description**
30319A	TurboSTORE/iX I	===>	N/A	Part of FOS
36387A	TurboSTORE/iX II	===>	B5151AA	TurboSTORE/iX II
36397A	TurboSTORE/iX II with Support for Optical Disk	===>	B5151AA	TurboSTORE/iX II
36388A	TurboSTORE/iX II with Online Backup	===>	B5152AA	TurboSTORE/iX 7x24 True-Online Backup
36398A	TurboSTORE/iX II with Support for Online Backup and Optical Disk	===>	B5152AA	TurboSTORE/iX 7x24 True-Online Backup

Important Details Please Read

For detailed information on all MPE/iX 5.0 TurboSTORE/iX products, refer to the *STORE and TurboSTORE/iX Manual* (30319-90001).

For detailed information on all MPE/iX 5.5 TurboSTORE/iX products, refer to the *STORE and TurboSTORE/iX Products Manual* (B5151-90001).

Some features are available in STORE/iX **and** in the TurboSTORE/iX II and TurboSTORE/iX 7x24 True-Online Backup products. The following table shows a comparison of the features in STORE/iX, TurboSTORE/iX II, and TurboSTORE/iX 7x24 True-Online Backup:

Feature Comparison

Feature	STORE/iX FOS	TurboSTORE/iX II B5151AA	TurboSTORE/iX 7x24 True-Online Backup B5152AA
Multiple Store Devices (`STORESET`)	YES	YES	YES
File Interleaving (`INTER`)	YES	YES	YES
Data Compression (`COMPRESS`)	NO	YES	YES
Parallel Restore (`RESTORESET`)	NO	YES	YES
Optical Device (`MOSET`)	NO	YES	YES
Online Backup (`ONLINE`)	NO	NO	YES
7x24 True-Online Backup (`ONLINE=START, END`)	NO	NO	YES
Store to Disk	NO	NO	YES
Restore from Disk	NO	NO	YES
Labeled Tapes	YES	YES	YES

Note

The TurboSTORE/iX 7x24 True-Online Backup product **does not** imply 24x7 (24 hours, 7 days a week) support. 7x24 represents the availability of the data that this product offers by not having to shut down business critical applications for backups. Customers should consult with their SE for the different levels of support with this product.

File System Resiliency Enhancements

by Frank Mendoza
Commercial Systems Division

Product Overview

The following enhancements for use by System Managers have been made to the MPE/iX 5.5 Release File System:

- System aborts have been reduced within the File System, and prevented when closing files on disk.
- A file *quarantine* mechanism has been added to handle files with corrupt data structures that cannot be closed.

  The FSCHECK utility was enhanced to include the following two new commands to support the file quarantine mechanism:

`QDISPLAY`	Displays information regarding file objects in quarantine.
`QPURGE`	Purges a specified file object in quarantine.

Important Details Please Read

The System Manager should have exclusive use of the system while running FSCHECK.

System Aborts

Some File System internal routines were modified to make use of a new operating system routine to assist in error processing. This feature allows the File System to retrieve more detailed information regarding traps and "escapes" enabling it to be more discriminating on the type of error that justifies calling system abort.

File Quarantine Mechanism

The file quarantine mechanism puts files with corrupt data structures in *quarantine*, restricting access to the file to read only. Files in quarantine can be handled in different ways, depending on the application. The following sections describe the different methods used to handle quarantine files.

Note

The file quarantine mechanism is active only if the Subsystem Dump facility is enabled. Subsystem Dump is off by default and must be enabled by running a privileged utility, SDUTIL.MPEXL.TELESUP. For more details, refer to the *Communicator* article, "Subsystem Dump Facility," in this chapter.

When a quarantine event occurs, a message is sent to the system console to alert the system manager. Each time a file is put in quarantine, relevant information regarding the event is written to the system log using record id *121* (decimal). This information may be displayed using the LOGTOOL utility.

Following is an example of system console messages:

```
WARNING!!!COULD NOT CLOSE FILE-
/YOURDIR1/YOURDIR2/YOURDIR3/----/YOURFILE.
FILE PUT IN QUARANTINE -
 Use FSCHECK command - QDISPLAY (QD) for more info.
or
WARNING!!!UNNAMED FILE PUT IN QUARANTINE BECAUSE OF CORRUPTION
UFID = xxxx xxxx xxxx xxxx
 Use FSCHECK command - QDISPLAY (QD) for more info.
or
WARNING!!! FILE SYSTEM RESOURCE CORRUPTION
GDPD ENTRY # xxxx PUT IN QUARANTINE.
```

The above messages are also instrumented to be output via the OpenView Console product. An application ID that associates the affected subsystem to a unique icon, in this case the MPE/iX icon, has been assigned. The color code indicates the severity of the message. The color of the icon is yellow, indicating there are warning events against the object.

If you get one of these messages, use the new FSCHECK utility command, `QDISPLAY`.

QDISPLAY Command

`QDISPLAY` displays information regarding file objects in quarantine.

Syntax

QDISPLAY [[DEV]= *All*]

Parameters

all	Displays quarantine information for each volume mounted in the MASTER or MEMBER state as reported by the MPE/iX `DSTAT` command.

Example

Following is an example of using `QDISPLAY`.

```
fscheck: QDISPLAY dev=all
```

```
<************ QUARANTINED FILE INFORMATION DISPLAY ************>

FILE NAME         : /a/b/c/d/f/e/badfile
UFID              : 055f0002 06cb0074 0009babe 0e031dd0 03447727
FILE LABEL        : 00000011.000e4570
GUFD              : 0000000a.ca00edfa
QUARANTINE DATE   : FRI, OCT 30, 1995, 11:33 AM
QUARANTINE REASON : dffdf006b

File Labels Allocated On Volume: MPEXL_SYSTEM_VOLUME_SET:MEMBER1
<**************************************************************>
      |
      |
<********* END OF QUARANTINED FILE INFORMATION DISPLAY ********>
```

Explanation of fields displayed:

FILE NAME	Fully qualified filename of file in quarantine.
UFID	Unique File Identifier of the file (20) bytes.
File Label	Virtual address of the file label.
GUFD	Virtual address of the Global Unique File Descriptor.
QUARANTINE DATE	Date and time the file was put in quarantine.
QUARANTINE REASON	Error status in MPE/iX format(info.subsys) that caused the file to be put in quarantine.

Methods for Handling Quarantined Files

Files in quarantine can be be handled using the following methods:

- Using the COPY command to create a new file with new data structures.
- Rebooting the system.
- Using the FSCHECK utility QPURGE command.

These methods are described below.

Using the COPY Command to Create a New File. Files in quarantine cannot be opened with write access. Therefore, they cannot be purged with the PURGE command. The file may be renamed to a different name and then copied back to the old filename using the COPY command. This method creates a new file with new data structures which is then accessible.

Rebooting the System. Schedule planned downtime and reboot the system to clean up and recover system resources used by files in quarantine. This method cleans up all corrupt transient data structures.

Using the FSCHECK Utility QPURGE Command. The FSCHECK utility command, QPURGE, purges a specified file object in quarantine.

Syntax

$$\texttt{QPURGE}\left\{\begin{array}{l}\texttt{Pathname = } \textit{pathname} \\ \texttt{UFID = } \textit{ufid}\end{array}\right\}$$

Parameters

pathname	Full pathname of the file to be purged.
ufid	Unique file identifier of the file object to be purged(40 hex digits).

Example

Following are examples of using QPURGE.

```
fscheck: QPURGE pathname=myfile.pub.sys or sys/pub/myfile

fscheck: QPURGE ufid="11111111 22222222 33333333 44444444 55555555"

fscheck: QPURGE ufid=11111111 22222222 33333333 44444444 55555555
```

Subsystem Dump Facility

By Walter McCullough
Commercial Systems Division

Product Overview

This article introduces the first release of the Subsystem Dump facility in the MPE/iX 5.5 Release. Subsystem Dump is part of the HP 3000 overall Software Resiliency strategy that is focusing on the avoidance of SYSTEM ABORT calls.

This release of Subsystem Dump can be thought of as a prototype of new software technology that focuses on online diagnosis by trained personnel. The focus on online diagnosis exists because of HP's commitment to decrease unplanned downtime.

What is Subsystem Dump?

Subsystem Dump is a facility that lets privileged subsystems *dump* the contents of a system failure into a Subsystem Dump file without aborting the system. You can then look at the file and find the error.

Enabling Subsystem Dump

When you boot up the system, the Subsystem Dump facility is disabled by default. Subsystem Dump is enabled by a privileged utility, SDUTIL in MPEXL.TELESUP. SDUTIL also creates and enables the SDUMP process, which is a process that manages the Subsystem Dump files.

To enable Subsystem Dump, use the SDUTIL `ON` command, either manually by entering the command interactively, or automatically by creating job files.

Manually Enabling Subsystem Dump

To manually enable Subsystem Dump, invoke the SDUTIL utility program and enter the `ON` command. SDUTIL prints out a header line and then prints out the Subsystem Dump status before it accepts any commands. See the `STATUS` command for an example of the Subsystem Dump status.

```
:SDUTIL.MPEXL.TELESUP
SDUTIL> ON
```

Other SDUTIL commands allow you to disable Subsystem Dump, see its status, or specify the size of a Subsystem Dump file. Following are the SDUTIL commands:

SDUTIL Commands

`HELP`	Displays and briefly describes all SDUTIL commands.
`ON`	Enables Subsystem Dump.
`OFF`	Disables Subsystem Dump.

`STATUS`	Provides the following information:

- Whether or not the SDUMP process exists.
- In which group and account Subsystem Dump files are being stored.
- The size of the dump files in sectors.
- The latest Subsystem Dump file that was used.

The following screen shows an example of the **STATUS** information.

```
SDUTIL> STATUS

SDUTIL A.00.01 (C) Hewlett-Packard Co., 1994.  All rights reserved.

---- STATUS ----
-SDUMP process    = ACTIVE
-SDUMP group      = SDUMP.TELESUP
-SDUMP file size  = 2000
-Last Dumpfile    = NONE
```

`SIZE`	Specifies the size of the Subsystem Dump files in sectors.

Note If the dump is too large for the file, or Subsystem Dump is disabled, a SYSTEM ABORT occurs.

`EXIT` or `QUIT`	Exits the SDUTIL utility.

Automatically Enabling Subsystem Dump

To set up your system to automatically enable Subsystem Dump, you may set up a logon UDC or create a job that executes through the SYSSTART file.

You can write command scripts to enable Subsystem Dump using the SDUTIL utility with the JCW facility. In the command script you can query whether or not Subsystem Dump is enabled or disabled.

Following is an example of a job stream that could automatically enable Subsystem Dump from the SYSSTART file:

```
:JOB SDUMP, MANAGER/pass.SYS/pass
:CONTINUE
:SDUTIL.MPEXL.TELESUP                  ON
:IF   JCW=0    THEN
:        TELLOP        SUBSYSTEM DUMP STARTED SUCCESSFULLY
:ELSE
:        TELLOP        SUBSYSTEM DUMP FAILED TO START!
:ENDIF
:EOJ
```

To see whether or not Subsystem Dump is enabled, you can use the `STATUS` command. If you see:

- `JCW=0`, then Subsystem Dump is enabled.
- `WARN0`, then Subsystem Dump is disabled.
- `FATAL0`, then the SDUMP process has terminated, which means it cannot create more Subsystem Dump files. Run SDUTIL with the `ON` command to restart the SDUMP process.

Subsystem Dump Files

A Subsystem Dump file is a structured disk file that contains limited system table data and information about only selected processes. This is in contrast to a full dump, which contains the entire contents of several system data structures and information about all processes on the system. When Subsystem Dump is enabled, a subsystem dump is taken automatically by a software subsystem when it detects a failure. This subsystem determines the actual contents of a dump.

The SDUMP process creates a pool of empty files to be used by Subsystem Dump. The pool files are named TDUMP000 through TDUMP099. When a subsystem, such as the File System, detects a failure, it dumps the failure information into a TDUMP file opened by the SDUMP process. This TDUMP file is then renamed to an SDUMP file with the same last three digits.

The TDUMP files are stored in the SDUMP group along with any Subsystem Dump files.

Opening and Analyzing Subsystem Dump Files with DAT

After the failure information has been stored, you can then open the file to analyze the problem using the `OPENSDUMP` command from DAT. For information on opening and analyzing the Subsystem Dump files, refer to the article, "Using DAT to Examine Subsystem Dumps," in this chapter.

Using DAT To Examine Subsystem Dumps

by Patrick Murphy
Commercial Systems Division

Introduction

A new facility called *Subsystem Dump* has been introduced with the MPE iX 5.5 Release. This article describes how you use DAT to analyze subsystem dumps.

A subsystem dump is a structured disk file that contains limited system table data and information about only selected processes. For more details, see the article, "Subsystem Dump Facility," in this chapter.

Dependencies with the System NL and SL Files

DAT uses the system NL and SL files for presenting information about processes in a dump. These files are required for stack tracing and displaying code executed by a process. Copies of these files are contained in full system dumps, but they are omitted from subsystem dumps to reduce the overall dump size. In the absence of these copies, DAT uses the current system NL and SL files instead. This means that if either of these files is to be updated before a subsystem dump is analyzed, you must make your own copies first.

Using STORE with Subsystem Dumps

When using `STORE` to save subsystem dump files offline, or to move them to different systems, the correct NL and SL files should be `STORE`d with them. SL.PUB.SYS requires special handling with `STORE`, since the CM Loader opens it with dynamic locking. The following `FILE` equation is required to `STORE` SL.PUB.SYS:

```
:FILE SL.PUB.SYS;LOCK
```

Opening Subsystem Dumps with DAT

Since subsystem dumps are created on disk, a `GETDUMP` operation is not required to access them. In addition, because subsystem dump files do not use the same naming convention as full system dumps (where MEM is appended to the dump name), the existing `OPENDUMP` also cannot be used to access them. Instead, the new command `OPENSDUMP` must be used to open subsystem dumps:

```
DAT> OPENSDUMP SDUMP009.SDUMP.TELESUP
```

If the subsystem dump is dependent on NL and SL files other than those currently in use by the system, `FILE` equations should be set up to aim DAT at the correct files before you use `OPENSDUMP`:

```
:FILE NL.PUB.SYS=MYNL
:FILE SL.PUB.SYS=MYSL
```

Navigating Subsystem Dumps

After a subsystem dump has been opened, all the familiar DAT commands may be used to examine the dump, provided the dump contains enough supporting information to allow the command to work.

For example, Format Virtual (FV) can be used with a symbol file to format dumped data structures. The PIN command can also be used to switch to the environment of a dumped process, whose stack may then be traced with the TR command. Note that these last two commands require access to the correct NL and SL files, as discussed earlier.

New DUMPINFO Output

As with full system dumps, the DUMPINFO command can be used to display subsystem dump contents. Use DUMPINFO ALL to display all characteristics of a dump.

The output of DUMPINFO MAP displays all virtual memory fragments contained in a dump. This may provide experienced users with useful clues about which data structures were dumped, and their specific locations.

A new option, PINS, may be used to obtain a list of all processes dumped, and their associated PIN numbers.

Introducing HP Patch/iX: A New MPE/iX Patch Management Tool

by Deb Alston
Commercial Systems Division

Overview

The MPE/iX Patch Management tool, HP Patch/iX, was introduced with PowerPatch C.50.05 based on MPE/iX General Release 5.0, and is available with the MPE/iX Release 5.5 Fundamental Operating System (FOS) software.

HP Patch/iX provides a user-friendly environment for customization and installation of Hewlett-Packard supported patches on an MPE system. The new tool provides the functionality of simultaneous installation of a PowerPatch tape with Reactive patches, thus reducing system unavailability associated with patch installations by as much as three hours per system. This downtime eliminates one of the system UPDATE/REBOOTs.

HP Patch/iX also provides qualification of all patches being applied to the system (both PowerPatch tapes and Reactive patches), thus reducing the likelihood of an older patch backing out a previously installed newer patch.

For more information, refer to the technical article, "HP Patch/iX Technical Overview," in Chapter 7, "Technical Articles." Detailed information on supported processes and comprehensive usage instructions are included in the *HP 3000 MPE/iX System Software Maintenance Manual Release 5.5* (30216-90223R3628).

Benefits and Features

HP Patch/iX allows:

- Qualification of all patches (Reactive and PowerPatch)
- Simultaneous installation of a PowerPatch tape with Reactive patches
- Customization of patches
- Creation of the patch installation tape while users remain on the system
- Creation of a CSLT only when necessary

HP Patch/iX Supported Processes

HP Patch/iX supports the following processes:

- Installation of a single Reactive patch
- Installation of multiple Reactive patches from various sources
- Installation of a PowerPatch tape
- Installation of a PowerPatch tape with Reactive patches

- Installation of a SUBSYS with a PowerPatch tape (referred to as an EXPRESS)
- Installation of a SUBSYS with a PowerPatch tape along with Reactive patches

---

Important Details Please Read

HP Patch/iX **does not** support the UPDATE of a system from one MPE release to another. A system installed with MPE/iX 5.0 **cannot** be UPDATEd to MPE/iX 5.5 via HP Patch/iX. The system must already be installed with MPE/iX 5.5 if patches are to be applied via Patch/iX. Additionally, HP Patch/iX does not support the addition of a SUBSYS product **without** a PowerPatch tape (SUBYS Add-On only), because there are required information files on the PowerPatch tape about the SUBSYS components.

Reflections Users

Reflections is a separately purchasable third-party product that is a PC-based terminal emulator program.

If you want to use Patch/iX from a PC using Reflections software, you must install Reflections with the `MPE POSIX` option (if available). Otherwise, Patch/iX will not work from your PC.

For more information on the Reflections `MPE POSIX` option, see the installation instructions for Reflections.

Introducing HP Stage/iX

By Michael Dovano
Commercial Systems Division

HP Stage/iX is a new operating system facility for applying and managing MPE/iX patches on your system. Using HP Stage/iX reduces system downtime and provides an easy and reliable method for backing out patches.

Use HP Stage/iX to place PowerPatch and/or reactive patches into staging areas on disk while the system is up. then choose a staging area to use at boot time to apply the patches. After the patches are applied. they can be backed out at any time through a reboot to the Base (the version applied by the last tape update). Once you are satisfied with the patches on the running system, you can commit the staging area to form a new Base while the system is running (no reboot is needed).

HP Stage/iX has three interfaces:

1. **HP Patch/iX menus** - Once HP Stage/iX is initialized, HP Patch/iX allows you to stage patches to staging areas (as well as create CSLT/STORE tapes in the usual fashion). Refer to the articles. "*Introducing HP Patch/iX: A New MPE/iX Patch Management Tool*." in Chapter 3. "System Management." and "*HP Patch/iX: Technical Overview*." in Chapter 10. "Technical Articles." for information about HP Patch/iX.
2. **The STAGEMAN utility** - This program allows you to manage your HP Stage/iX environment. and obtain information about the environment and individual staging areas.
3. **The STAGEISL utility** - This is an ISL utility available when the system is down. It contains a subset of the STAGEMAN functionality. and allows you to recover from most errors or mistakes.

HP Stage/iX Concepts

Your operating system resides in what HP Stage/iX refers to as the **Base**. This is the set of files laid down by the last system installation or update (from tape). HP Stage/iX creates and manages **staging areas**. which are file "containers" on disk that hold versions of files that are different from the Base (a staging area is actually an HFS directory which holds all the files associated with that staging area). More than one staging area can exist at a time. Each staging area contains the difference. or delta. between the Base Operating System and a patched OS.

When a staging area is **activated** on the next boot. the files in the staging area's directory are moved (renamed) into their **natural locations** (for example. the staged version of the NL is moved into NL.PUB.SYS). At the same time. the Base versions of the files are saved into an HP Stage/iX archive directory. When the staging area is backed out (when the system is booted back to the Base), the converse takes place and the system is restored to its original state.

When an active staging area is **committed** to the Base, the staging area's directory is deleted, and all archived Base files are purged. The files that were switched into their natural locations when the staging area was activated remain there as part of the new Base. This releases any disk space that was used by the staging area.

HP Stage/iX (with the help of HP Patch/iX) allows new patches to be staged and applied in a cumulative fashion. This means that if you create a new staging area while a staging area is active, the new staging area will contain all the changes between the Base and the active staging area, *plus* the new patches applied to the new staging area.

Requirements

Since putting patches in staging areas requires that additional files be placed on disk, using HP Stage/iX tends to use a lot of disk space. Adequate disk space on LDEV1 is especially critical, since a large percentage of patches require changes to large files that must be staged on LDEV1 (`NL.PUB.SYS` is an example). Because of this, HP Stage/iX is targeted for larger HP3000 systems with adequate extra disk space.

Installing and Initializing HP Stage/iX

HP Stage/iX is part of the Fundamental Operating System (FOS), and is therefore automatically installed when you update to OS version 5.5 or later. To be able to use HP Stage/iX you must:

1. Update or install your system software to version 5.5.

   Refer to the *HP 3000 MPE/iX System Software Maintenance Manual Release 5.5* (30216-90223R3628) and follow the directions in the manual for updating and/or installing your system software.

2. Install HP Patch/iX.

   Refer to the *HP 3000 MPE/iX System Software Maintenance Manual Release 5.5* (30216-90223R3628) and follow the directions in the manual for installing HP Patch/iX and applying patches using staging areas. Installing HP Patch/iX is included as part of the "Manage Patches by Staging Area" task.

   This step is required to manage your qualified patches.

3. Initialize HP Stage/iX using the `STAGEMAN` utility.

   Refer to the *HP 3000 MPE/iX System Software Maintenance Manual Release 5.5* (30216-90223R3628) for initialization instructions.

HP Stage/iX HELP

Help is available for all HP Stage/iX functions from within the HP Patch/iX, `STAGEMAN`, and `STAGEISL` facilities. With `STAGEMAN` or `STAGEISL`, use the `HELP` command. With HP Patch/iX, help is available in HP Stage/iX context with the F1 function key.

4

POSIX/Open Solutions

Transferring HFS-Named Files With MPE Attributes (MOVER Utility)

by Steve Elmer
Commercial Systems Division

Beginning with the MPE/iX-Express 2 based on General Release 5.0, a new TELESUP utility, MOVER.PRVXL.TELESUP, is available. The MOVER utility is essentially an upgrade to the UHAUL utility that allows HFS files to be archived while providing expanded features. The utility is documented in MOVER.DOCXL.TELESUP.

Prior to the Express 2 Release, transferring files via the network has posed many difficulties when using HFS names for files. There have also been problems with new file formats and preserving MPE file attributes.

Each of the existing archive utilities falls short of the MOVER utility for the following reasons:

- The shell's tar utility can handle HFS names and the new file types, but drops the MPE file attributes.
- The STORE command does not support creating a disk file, so even though it handles all the problems, it cannot be used to transfer files over a network.
- The UHAUL utility in TELESUP can handle the new file types and all the MPE file attributes, but cannot handle the HFS name space.

MOVER Features

The MOVER utility combines and compresses multiple disk files into a single disk file. This file can then be easily transported to another system through products like NS/3000 or the ARPA FTP program. Once transported the MOVER utility can again be run to reconstruct the original files.

Following is a list of the MOVER features:

- Keeps many related files together in one easy to transport location (like patch files and their job streams).
- Reconstructs files exactly as they were originally created including:
  - File type (including symbolic links)
  - File code
  - File limit

    - Record size
    - Record format (including byte-stream)
    - Blocking factor
    - Extent size
    - Number of extents
    - Number of user labels
    - User label contents
    - Security information (as POSIX user/group/other rwx bits)
- Compresses and simplifies moving multiple files from one HP 3000 to another. (ASCII data is anywhere from 30-80% compressible. Binary files are usually about 5-15%.) This data compression can reduce data transmission times.
- Uses the byte-stream record format and has no dependencies on MPE file attributes for the archive file itself.
- Recognizes directories and automatically recurses them.
- Uses a LISTF-like format to display file information for every file archived.
- When privileged, can archive some common database files.
- Supports both relative and absolute filename modes.

  To archive files with only relative names, simply use relative names in the fileset. To archive files with absolute names, use absolute names in the fileset.

  For example, if the Current Working Directory (CWD) is "/" then the command

  ```
  MOVER -c foo .
  ```

  archives every file on the system using relative names. The command

  ```
  MOVER -c foo /
  ```

  would archive every file on the system using absolute names. Files archived with relative names can be unarchived into any directory. Files archived with absolute names are unarchived only to their original filename.

MOVER Syntax

The MOVER program is a POSIX program and takes a POSIX syntax. When you use the shell to invoke MOVER it expands filename wildcards appropriately for MOVER's use.

To create a new archive, use

```
MOVER -c archive_name file file ...
```

To extract files from an existing archive, use

```
MOVER -x archive_name
```

To list the files in an existing archive, use

```
MOVER -t archive_name
```

To allow MOVER to manipulate privileged files, add "0" to the options. For example, the following command

```
MOVER -c0 foo
```

would archive privileged files in the CWD.

MOVER Safety

To remove PM capability from MOVER, use the linkeditor:

```
:linkedit
linkedit/ix> altprog MOVER;cap=ia,ba
```

The utility still functions, but the "0" option is ignored.

The MOVER utility overwrites files by default. Use care when unarchiving files, especially absolute named files.

CI Enhancements Overview

by Jeff Vance
Commercial Systems Division

Overview

The Command Interpreter (CI) has been improved in MPE/iX Release 5.5, which includes the CI enhancements added in MPE/iX-Express 3 based on General Release 5.0.

This article briefly summarizes the enhancements. For more details, see the article, "A Detailed Look at CI Enhancements," in Chapter 10, "Technical Articles."

CI Enhancements

The following enhancements were added to the CI in MPE/iX Release 5.5:

- Eight new predefined CI variables) were added:

  ```
  HPLOCIPADDR
  HPREMIPADDR
  HPLOCPORT
  HPREMPORT
  HPSTREAMEDBY
  HPLASTJOB
  HPOSVERSION
  HPRELVERSION
  ```

  Now it is possible to filter remote connections to your HP 3000 via a logon UDC that examines the remote accessor's IP address and port number. This information is also available via the AIFs (please see the *Communicator* articles: "AIF Enhancement to AIFPROCGET Overview," in Chapter 3, "System Management," and "AIF Enhancement to AIFPROCGET Details," in Chapter 10, "Technical Articles").
- To support the longer HFS names, CI string variables are now 1024 characters long. Additionally, approximately twice the number of user-defined variables can be created in MPE/iX Release 5.5.
- Parsing of user input and command output is easier in Release 5.5. Five new CI evaluator functions were added: `word()`, `edit()`, `repl()`, `delimpos()` and `pmatch()`. Also the `input()`, `str()` and `rht()` functions have been enhanced.
- Command files and UDCs can be protected by granting only execute (X) access to their users. You can execute the command file but cannot see its contents. This is the same behavior that has been available for stream files.
- To ease password management, the `JOBSECURITY` command was enhanced to allow jobs to be streamed without embedded passwords and without password prompting. This is described in

the *Communicator* article, "New JOBSECURITY Feature," in Chapter 3, "System Management."

- Command files can use the full POSIX naming standards, and the `HPPATH` variable now supports directory names.
- The `REDO` command can upshift and downshift characters and delete, upshift or downshift a word.
- The CI recognizes a leading "#" as a comment line.
- Online help is now available for all CI variables and evaluator functions and all `FINFO` items. Simply enter:.

`:help VARIABLES`	To see all of the predefined CI variables
`:help HPCWD`	(or any variable name) to see a description of an individual variable
`:help FINFO`	To see the FINFO items

Exec Enhancement

by Steve Elmer
Commercial Systems Division

Enhancement Overview

The *exec* functions are used to replace the currently running program with a new program, within the same process. This can be contrasted with the CREATEPROCESS intrinsic which has two parts: create a new process, and load a new program for that process. The *exec* functions do only the last part—load a new program onto the calling process. The old program "disappears" and is never returned to.

Prior to MPE/iX Release 5.5, the *exec* functions could only load programs. In MPE/iX Release 5.5 the *exec* functions have been enhanced to load some non-program files, called *exec scripts*.

An exec script is a text file containing data to be interpreted by another program (after all, **some** program must be loaded.) Whenever the "program" to be loaded by *exec* is not a file with the NMPROG filecode, it is examined to see if it is an exec script. An exec script has a special first line that contains the name of the interpreter program appropriate to this script.

Once *exec* has the script name and the interpreter name, it simply loads in the interpreter and passes the original script name to it as an argument. The interpreter then opens the script file, reads its contents, and interprets them. As an example, if the interpreter were the POSIX shell, then the data would be shell commands.

Features and Benefits

Following are the features and benefits of the *exec* scripts enhancement:

- This enhancement treats interpreted scripts as though they were program files. This is a helpful abstraction that reduces the overhead of managing script files and simplifies the use of interpreted scripts in applications.
- Since the script contains the interpreter name, the calling program doesn't have to hard-code the interpreter-to-script relationship. If the script is re-implemented with another interpreter or as a program, the calling program does not have to change.
- The most important benefit is the ability to easily port existing scripts from Unix to MPE/iX. As with POSIX itself, porting is greatly simplified. In many cases, the script will operate correctly, without modification.

What is an Exec Script?

An exec script is a file containing a reference to an interpreter program and commands to be executed by that program. When one of the POSIX *exec* interfaces is used to load the exec script file, the interpreter program is loaded instead, and given the script name as a parameter. The interpreter program then opens and reads the script file taking whatever actions are specified.

Many programs on the system are actually interpreters. These programs take data files as input that directs them to perform certain actions. The POSIX shell is an example of an interpreter that reads script files containing shell commands and executes those commands. Some other common examples of interpreters include: awk, grep, and perl.

If you use Unix systems, you may have been using exec scripts without realizing it. Most shell scripts have a first line such as:

```
#! /bin/ksh
```

When "#!" are the first two characters in the script file, the remainder of the line is the interpreter program's name and any arguments for the program. In this case, the "/bin/ksh" program is the interpreter and there are no arguments. Since the "#" in the first character is a comment character for the shell, it is ignored. The remainder of the file is executed by the shell.

In general, interpreters that exist on both Unix and MPE are valid interpreters for an exec script. This allows for simple migration from Unix to MPE, generally with no changes required in the script file.

Note

Not all interpreter programs can successfully take advantage of the exec scripts feature. The CI is an example of such an interpreter. Since the parameter passing mechanism is geared toward POSIX programs, the CI is unable to determine what script to execute. Some experimentation may be required to determine whether a program can successfully be invoked as an exec scripts interpreter.

Exec scripts allow a new level of abstraction for the programmer. Since the script file can now name its own interpreter, the calling program can invoke the script as if it actually **were** a program! Without exec scripts, this would be a difficult and error prone programming task in the general case.

Exec Scripts Specification

This section contains the technical information necessary to understand and implement exec scripts. Knowledge of C programming will help you understand this material better.

C Programming Background Information

The operating system passes C programs an array of strings known as the argument list. This array is conventionally given the name argv. In your C program it typically looks like this:

```
main( int argc, char *argv[] )
```

or

```
main( int argc, char **argv )
```

C arrays typically begin with the index zero (0). Thus, *argv[0]* is a pointer to the first argument string. The *argc* parameter tells you how many elements are in the *argv* array.

```
+--------+
|   0    |------->"testprog"
+--------+
|   1    |------->"-x"
+--------+
|  ...   |------->"other arguments"
+--------+
| argc   |------->"last argument"
+--------+
```

Script File Format

The first line of the script file must have the form:

```
#! interpreter [argument]
```

The *interpreter* must be an absolute POSIX pathname. The *argument* is an optional string of characters that is passed as a single argument to the *interpreter*. Spaces and tabs between the "#!" and the *interpreter*, and between the *interpreter* and the *argument* are ignored. After leading whitespace is trimmed, the entire argument string is gathered up to the '\n' or the 1024 character limit to form the *argument*.

Argument Passing

The argument list from the *exec()* call is passed along to the interpreter with modifications. When the *argument* is present, it is added to the argument list for the interpreter. Whether the *argument* is present or not, the script filename is added to the argument list.

The *exec()* call may have arguments as well. By convention, *argv[0]* always contains the new program's name. This argument is passed along to the interpreter unchanged. Any remaining arguments are shifted to make room for the Exec Scripts arguments.

When the *argument* is not supplied, a new argument is inserted between *argv[0]* and *argv[1]* and contains the name of the script to be interpreted. When the *argument* is supplied, two new arguments are inserted between *argv[0]* and *argv[1]* - first the *argument* value, and then the script name.

An example of each form:

```
:cat print_args.c
/* program to just print the args */
/* passed in and exit             */
main (int argc, char *argv[])
{
  int i;

  for (i=0;i<argc;i++)
    printf("%d: '%s'\n", i, argv[i]);
}
```

The following script invokes the program without interpreter arguments. Its output is shown below.

```
:cat myscript
#!print_args

:myscript 1 2 3
0: 'myscript'
1: './myscript'
2: '1'
3: '2'
4: '3'
```

The following script invokes the program with interpreter arguments. Its output is shown below.

```
:cat myscript2
#!       print_args       arg0 arg1 arg2
#  ^- spaces ignored -^

:myscript2 1 2 3
0: 'myscript2'
1: 'arg0 arg1 arg2'
2: './myscript2'
3: '1'
4: '2'
5: '3'
```

Exec Script Requirements

A valid script must meet the following requirements:

- The file must be a byte-stream file or an emulatable byte-stream file.
- The first two characters of the file must be "#!".
- The interpreter name must be given as an absolute path.
- The interpreter name must reference a valid program file.
- There may be zero or more spaces or tabs between the "#!" and the interpreter name.
- An optional argument list may follow the interpreter name.
- If an argument list is given, one or more spaces or tabs *must* separate the interpreter name and the argument list.
- The entire string must be terminated by a null or newline character.
- The terminator character delimits the argument list.
- The entire string inclusive of the terminator character may not exceed 1024 bytes (extending the Unix 32-byte minimum.)

The argument list is modified by *exec()* before initiating the interpreter.

- The 0th argument remains unchanged from the *exec()* call as it specifies the naming convention.
- If there were no *exec()* arguments, a phony *argv[0]* is created to point at a null string.
- If an interpreter argument list is present:
  - The *exec() argv[0]* remains unchanged.
  - The interpreter argument list from the script is inserted into the *exec()* arguments as a new *argv[1]*.
  - The script name is inserted into the *exec()* arguments as a new *argv[2]*.
  - The remainder of the *exec()* arguments are shifted to begin at *argv[3]*.
- If the argument list is not present:
  - The *exec() argv[0]* remains unchanged.
  - The script name is inserted into the *exec()* arguments as a new *argv[1]*.
  - The remainder of the *exec()* arguments are shifted to begin at *argv[2]*.

The set-user-id and set-group-id bits are ignored for script files. The interpreter's set-user-id and set-group-id bits are honored.

POSIX Shell Enhancements - lp and tar Commands

by Kevin Cooper
Commercial Systems Division

Overview

The following two pieces of new POSIX shell functionality (requested by software vendors porting to the HP 3000), are now available in the MPE/iX 5.5 Release:

- Several new commands (`lp`, along with some associated commands), have been added for printing.
- A major enhancement to the `tar` command (Tape ARchiver), has been added so that `tar` can handle many MPE/iX file types, and be usable for software distribution on the HP 3000.

lp Command

The `lp` enhancement provides the following four new POSIX shell commands:

`lp`	Prints a file.
`lpstat`	Shows the status of spool files.
`lpalt`	Allows you to alter some characteristics of spool files.
`cancel`	Cancels a print request.

These commands have been implemented using the MPE/iX native mode spooler. They allow you to print files from within the shell, and control your spool files after they have been sent to the spooler.

tar Command

The `tar` command has been enhanced specifically to support retention of MPE/iX file attributes, both when creating an archive and when the file is later recreated from the archive.

For example, `tar` now handles NMPRG files that are fixed, binary (rather than just byte-stream) files. Nine MPE/iX attributes are now stored along with the file, and this is done according to the proposed POSIX.2b standard format for `tar`. The saved attributes are:

- Record Size
- File Code
- File Limit
- Number of Extents
- Number of User Labels
- Blocking Factor
- Record Format (Fixed, Variable, Byte-Stream, etc.)
- Carriage Control
- ASCII or Binary

If the file contains user labels, these are also saved in the archive and recreated on disk later by `tar`.

The `tar` command is not designed to be a backup tool, since there are some file types that it does not handle. This enhancement was designed to give software vendors who port using POSIX a way within the shell to distribute their software.

Types of files that `tar` cannot handle include:

- PRIV files, such as databases
- Files that are MPE-specific, such as MSG, CIR, and RIO files

`Tar` skips over these files, prints a message, and continues operation.

More information on these new and enhanced shell commands is available in the online man pages distributed with MPE/iX Release 5.5.

MPE/iX POSIX Developer's Kit Bundled with Release 5.5

by Joseph Feiner
Commercial Systems Division

Overview

With MPE/iX Release 5.5, the MPE/iX Developer's Kit will be bundled with the core operating system for all users of the HP 3000. However, the HP C/iX product (HP product number HP31506), must be purchased to compile POSIX.1 source. Previously, the MPE/iX Developer's Kit was a no-charge upgrade when you purchased the C/iX compiler, but you had to order the free MPE/iX Developer's Kit separately.

The following software was in the MPE/iX Developer's Kit:

- AT&T SVID Interprocess Communication (IPC) API
- X/Open Curses API
- /lib/libc.a—the developer's RL for access to the Posix APIs.

This software allows HP 3000 developers access to POSIX 1003.1 interfaces, POSIX 1003.2 commands and utilities, as well as other industry-standard operating system features. Customers can create POSIX-compliant applications built for open systems environments on their HP 3000 machines. Application developers can, in conjunction with this software, develop new, POSIX-compliant, open systems applications for their HP 3000 systems using C, C++, or Micro Focus COBOL.

This software is designed for customers who want to develop POSIX applications or port POSIX applications to the HP 3000. The kit includes software for application developers.

Users are provided a relocatable library (RL) that allows access to the necessary C language interfaces. Once the object files have been linked with this RL, end-users can run the resulting POSIX application on an HP 3000 Series 900 system.

Since the MPE/iX Developer's Kit (HP35430) is bundled with the core operating system, it is no longer orderable as a separate product.

Documentation

Consult the following documents for more details on the operation of the components that comprise these newly included components:

- *HP C/iX Reference Manual (31506-90005)*
- *HP C Programmer's Guide (92434-90002)*
- *HP C/iX Library Reference Manual (30026-90001)*
- *MPE/iX Developer's Kit Reference Manual Vol 1 & 2 (36430-60001)*

- *The POSIX.1 Standard: A Programmer's Guide* (ISBN 0-8053-9605-5)
- *MPE/iX Shell and Utilities Reference Manual Vol 1 & 2* (36431-60001)
- *MPE/iX Shell and Utilities User's Guide* (36431-90002)

Customers who purchase HP C/iX on MPE/iX Release 5.5 or later will automatically receive these manuals with their order. Manuals may also be purchased separately.

5

System Support Tools

Predictive Support for New Peripherals and SPUs

Mike McNelly
Worldwide Customer Support Operations

Product Overview

HP Predictive Support provides proactive hardware support and helps increase the uptime of your systems by monitoring system memory and disk/tape drives.

When the HP Predictive Support software detects a potential problem, it sends a message to the HP Response Center. The Response Center portion of the system screens the data and forwards problems requiring further analysis to a Response Center Engineer. If action is needed at your site, the Response Center Engineer and the account Customer Engineer will work with you until the problem is resolved.

This proactive hardware support is provided as part of the HP Hardware and Software Support Agreement.

Predictive Enhancements for MPE/iX Release 5.5

Predictive has added autoconfig functionality which **automatically** configures in **all** disks and tapes on a system. It does this by using the mechanism serial and product numbers returned from an inquiry command to the disk. This new functionality eliminates the AUTOCONFIG(450) and AUTOCONFIG(545) events and it should significantly reduce installation time and effort for new installations or when installing or replacing peripherals.

Predictive has also added enhanced scheduling ("Real Time") functionality which allows more scheduling options for Predictive runs. You can specify that the Predictive scanners (MEMSCAN, LOGSCAN, DISCSCAN, and SCSISCAN) should run more than once a day at intervals between 15 minutes and 12 hours, and you can specify a different schedule for each of the scanners. You can also specify a range of time when Predictive should not run (such as during backups).

Upgrading to MPE/iX Release 5.5

Since HP Predictive Support runs as a system process, interacting extensively with the diagnostic subsystem to detect errors, it is *essential* that HP Predictive Support be updated along with the MPE/iX Release 5.5 (C.55.00) Fundamental Operating System (FOS).

Support for Hewlett-Packard's New Hardware Platforms and Peripherals

HP Predictive Support supports the following new hardware platforms and peripherals:

- 969KS/100-400 120 MHz SPU
- 996/80 and 996/100-800 SPU
- A2958A 1 Gbyte SCSI Disk drive
- A3191A 2 Gbyte SCSI Disk drive
- A3304A,A3351A 2 Gbyte SCSI Disk drive
- A3352A,A3353A 4 Gbyte SCSI Disk drive
- HPC1536A DDS Tape Drive

Before You Update HP Predictive Support

Before you install the update, you should run HP Predictive Support to ensure that any data indicating potential problems is transmitted to the HP Response Center.

If you have an HP Hardware and Software Support Agreement, and wish to have HP Predictive Support installed, please contact Hewlett-Packard and a Customer Engineer will perform the initial installation.

6 Application Development

Introducing Dependent Libraries on MPE/iX Loader

by Huong Ho and Ed Olander
Commercial Systems Division

Overview

Beginning with MPE/iX Release 5.5, MPE/iX allows executable libraries (XLs) to specify dependent XLs to be implicitly loaded for symbol binding and resolution purposes. This new functionality is referred to as *Dependent Libraries*. It gives HP 3000 users an extended loading capability to include data as well as code.

Executable libraries are enhanced to recognize and process the library dependency list so that a library can cause other libraries to be implicitly loaded. This, in effect, allows dynamically loaded libraries to bind their code symbols through already loaded libraries.

All these capabilities bring MPE/iX executable libraries even closer to HP-UX shared libraries and make porting products from HP-UX to MPE/iX even simpler.

Intended Audience

This article is intended for system programmers, system managers, and individuals interested in writing application programs or in porting industry standard applications. The article provides a general overview of Dependent Libraries along with their features and benefits.

Features and Benefits

Modern application development environment encourages application developers to employ runtime libraries for greater efficiency and flexibility. Hence, runtime or executable libraries must be effectively managed and organized. Dependent Libraries help make MPE/iX executable libraries easier to manage and use by allowing dynamically loaded libraries to be loaded through a multiple of entry points in the loader binding sequence. The features and benefits of Dependent Libraries can be briefly described as follows:

Features

- **Extensibility** - Dependent Libraries provide a simple design approach that allows a list of libraries to be expanded and constructed through a specification of a *single* library. CM applications can now be switched over to NM and resolved through an expanded, unfolded list of Dependent Libraries.

- **Portability** - Dependent Libraries on MPE/iX resemble HP-UX Shared Library Dependencies. This facilitates and reduces the porting effort of UNIX-based applications to the HP 3000 which increases the application availability on the HP 3000.
- **Transparency** - The enhancements to MPE/iX are highly transparent to the users. You can still compile, link and run your applications as before. Previously linked executables and compiled scripts continue to work on MPE/iX Release 5.5 and later.

Benefits

- **Flexibility** - The ability to specify a dependency list in the firstfile library adds flexibility to the Compatibility Mode (CM) Switch-Stub to Native Mode (NM) binding path. It allows a developer to write a CM->NM switch stub that invokes a procedure in one XL that subsequently calls another procedure in another XL.
- **Supportability** - Dependent library usage at runtime provides an alternative mechanism to achieve multiple binding of XL libraries that is currently not allowed in the libname parameter of the CM `HPSWTONMNAME` intrinsic. This enhancement helps increase the supportability of `HPSWTONMNAME` by removing its current major limitation.
- **Maintainability** - Channel Partner developers can now utilize the Dependent Library to repackage and improve their applications delivery to their end-users while maintaining minimal user intrusion and interaction. All library dependencies and resolutions are hidden from the end-users.
- **Simplicity** - A whole binding tree can be built and expanded through a *single* parameter, *firstfile*, to the `HPGETPROCPLABEL` intrinsic and yet there are no new interfaces or external changes required by the loader.

Please refer to the "MPE/iX Dependent Libraries Technical Overview" article in Chapter 10, "Technical Articles," for more detailed information about Dependent Libraries and their related components. Also refer to the following general and technical articles, "HP Link Editor/iX Enhancements Overview" in this chapter, and "HP Link Editor/iX Enhancements Detail" in Chapter 10 for additional Link Editor/iX functionality made available because of Dependent Libraries.

HP Link Editor/iX Enhancements Overview

by Sue Meloy
Support Technology Center

Support for Dependent Libraries

The HP Link Editor/iX has been enhanced to support Dependent Libraries introduced in this MPE/iX 5.5 Release. A Dependent Library is a list of dependent executable libraries (XLs) that are loaded in addition to the executable library that specified them. Refer to the following articles for more details on Dependent Libraries:

- "Introducing Dependent Libraries on MPE/iX Loader" in this chapter.
- "MPE/iX Dependent Libraries Technical Overview" in Chapter 10.

HP Link Editor/iX Changes

The Dependent Library list can be manipulated using the HP Link Editor/iX new command, `ALTXL`, and the enhanced command, `BUILDXL`. `ALTXL` allows you to change the dependent library string for an executable library. `BUILDXL` adds another option for specifying the dependent library string.

Refer to the "HP Link Editor/iX Enhancements Detail" technical article for details on these two commands.

RPG/iX Enhancements for Version A.00.14

by Don Jenkins
Worldwide Response Center Organization

Introduction

This article describes three new enhancements to RPG/iX A.00.14, released on MPE/iX Release 5.5. These include:

- User access to the VPlus COMAREA
- Retaining the binary format of file numbers using the FNUM operator
- Additions to the DSPLY operator

VPlus COMAREA Access

This enhancement allows user access to the VPlus Communication Area (COMAREA), for both reading and modifying. Users may now modify various actions performed by RPG, or insert calls to the various VPlus intrinsics as desired, using the intrinsic calling mechanism of RPG/iX.

In the past, this area was managed only by RPG, which was consistent with the philosophy of making all lower-level I/O transparent to the user. One problem with this approach is that for certain errors on VPlus intrinsic calls, the application aborts with no possibility of recovery. With terminals and printers now being attached to systems over networks, where noise may be a problem, this is no longer acceptable.

The new enhancement is enabled for your application by placing a $CONTROL VPLUSCOM record prior to your H-specification record. If you make no other changes to your program, it continues to run as before, with the exception of certain errors on return from VPlus intrinsic calls (COMAREA Status word not equal to zero). In general, these errors are on intrinsic calls that access the terminal or printer, and are discussed in more detail later. If an error does occur in this case, your program does not abort, and strange things could happen. To take care of this situation, read on.

When the RPG/iX compiler detects the statement, `$CONTROL VPLUSCOM`, it sets a flag used to control certain items, both at compile-time and at run-time. RPG/iX first creates two new reserved-word variables, "*VC" and "*VSTAT". *VC is declared to be a 200 character alphanumeric array of 1 character per element. Do NOT enter an E-spec for this variable. *VSTAT is declared to be a 6 digit numeric variable with 0 decimal places.

Note

If you try using these items without the `$CONTROL` statement, you get a compile-time error.

Since these are declared for you, all you need to do is use them. No other declarations are necessary.

The VPlus COMAREA is specified to be either 60 or 85 16-bit words in length (see Chapter 6 (pp 6-20) in the *Data Entry and Forms Management System VPlus Reference Manual* (32209-60002). It contains a mix of binary integer, double integer, logical, and ASCII data. RPG/iX allocates a space of 100 16-bit words for the VPlus COMAREA, located in a special area of the run-time data space (the additional space is there just in case VPlus ever expands the COMAREA).

The VPlus COMAREA is zeroed out prior to running a VPlus application. When RPG/iX calls **VOPENTERM** to begin everything, it sets:

1. The COMAREA length (word 3) to 85 (this is arbitrary, but allows 3075/6 terminals to be used).
2. The language (word 2) to 5 (this specifies PASCAL, but works equally well for C, in which RPG/iX is written).
3. The `label'option` (word 10) to the value specified in the workstation F-spec column 50.
4. The `form'store'size` (word 39) to the value specified in the workstation F-spec FORMSDL continuation record, if used.

The new reserved word *VC is initialized by the compiler to point to the start of the COMAREA. Thus, *VC.1 and *VC.2 point to the left and right bytes of the status word. Since RPG does not have an internal binary data type, managing the COMAREA by the user can be somewhat tricky. This is discussed later.

The reserved word *VSTAT may be used as an ordinary 6-digit numeric variable, but has special properties. For certain actions (**SHOMSG**, **SHOW**, **RDTERM**, **SHODATA**, **PRINTX**, **CLRMSG**), when the VPlus intrinsic is called, the COMAREA status word is copied into *VSTAT. The COMAREA status word (the first 16-bit word of the COMAREA) contains the result of a VPlus intrinsic call. If the intrinsic executes correctly, this value is zero. If it fails, the value returned is a binary integer equal to an error number related to the reason for the failure. It is your responsibility to test *VSTAT immediately after the **EXCPT** statement that initiates the action. See Example 3, shown later, for an example of testing *VSTAT.

If *VSTAT is non-zero, you must set it to zero before taking any other action (except possibly calling VERRMSG using INTR/IPARM operators—see Example 1, shown later, for an example of setting *VSTAT). To reset the status word to zero do:

```
Z-ADDO          *VSTAT
```

This sets both *VSTAT and the COMAREA status word to zero. ONLY the Z-ADD operator does this. If you Z-ADD a non-zero value to *VSTAT, *VSTAT is set to this value but the COMAREA status word is still set to zero.

As you may surmise from the above, you may now call VPlus intrinsics directly from your RPG program using the INTR/IPARM operators. When the intrinsic is called, RPG/iX updates *VSTAT with the COMAREA status word. You must test this yourself to verify correct operation. This may only be done if you have specified $CONTROL VPLUSCOM in your program which is a workstation application. The intrinsic begins with the letter "V" (a potential limitation, but is OK for now).

If HP releases a new intrinsic beginning with "V" and you use it in your VPlus application, *VSTAT would still be updated from the status word in the COMAREA, but it would be meaningless for the new call (unless, of course, it is a new VPlus intrinsic)—ignore it.

Due to the philosophy of managing lower-level I/O for users, RPG/iX takes care of calling VOPENTERM and VCLOSETERM. Since these intrinsics also communicate with the terminal, it is possible they can also fail, with the resultant abort. With this enhancement, RPG now tests the status word returned from the call, and if the status word is non-zero, RPG retries the intrinsic up to 5 times, with a one second pause between each retry, before aborting. This is done regardless of whether or not you have specified the $CONTROL VPLUSCOM statement. It is also done for the intrinsic VSHOWFORM where it is called from certain error routines not accessible to users.

Some VPlus intrinsics that fail are not recoverable, i.e., there is some system failure that caused the problem. The response to these failures has not been modified. A program abort may still occur if the status return from an intrinsic call is non-zero.

Following is a list of the RPG actions that have been modified when using the $CONTROL VPLUSCOM statement, the intrinsics called by the actions, and the behavior of the enhancement:

SHOMSG	Calls VPUTWINDOW, VSHOWFORM. If VPUTWINDOW fails, it puts the error message in the window and still calls VSHOWFORM. After VSHOWFORM is called, RPG copies the COMAREA status word into *VSTAT. If the status word is non-zero, an error message is placed in the window and you are returned to the program with no abort. You must test *VSTAT and decide what to do. WINDOWENH in the COMAREA may also be modified as in PUTMSG described below.

`SHOW`	Calls `VSHOWFORM`. Modifies COMAREA item `SHOWCONTROL` bit 9, depending on the value in the output buffer column 7 (blank or `P`—the preload option). After `VSHOWFORM` is called, RPG copies the COMAREA status word into *VSTAT. If the status word is non-zero, an error message is placed in the window and you are returned to the program with no abort. You must test *VSTAT and decide what to do.
`RDTERM`	Calls `VREADFIELDS`. The COMAREA item `LOOK'AHEAD` may be modified depending on the value in the output buffer, column 8. After `VREADFIELDS` is called, RPG copies the COMAREA status word into *VSTAT. If the status word is non-zero, an error message is placed in the window and you are returned to the program with no abort. You must test *VSTAT and decide what to do.

Note

The *HP RPG/iX Reference Manual* (30318-60002) (page 13-6), notes that `RDTERM` also calls `VGETBUFFER`. This is actually done by the `READ` operation following the `RDTERM` action. There is no change from the previous behavior for this call.

`CORERR`	Calls `VSETERROR`, `VSHOWFORM`, and `VREADFIELDS`. If `VSETERROR` fails, it puts an error message in the window and continues with `VSHOWFORM`. After either `VSHOWFORM` or `VREADFIELDS` have been called, RPG copies the COMAREA status word into *VSTAT. If there is no failure, the final *VSTAT value is from `VREADFIELDS`. If either fails, an appropriate error message is placed in the window, and you are returned to the program with no abort. In any case, it is your responsibility to test *VSTAT and decide what to do.
`SHODTA`	Calls `VPUTBUFFER`, `VSHOWFORM`. If `VPUTBUFFER` fails, it puts an error message in the window and continues with `VSHOWFORM`. After `VSHOWFORM` is called, RPG copies the COMAREA status word into *VSTAT. If it is non-zero, an error message is placed in the window and you are returned to the program with no abort. You must test *VSTAT to decide what to do.
`PRINT`	Calls `VPRINTFORM`. After `VPRINTFORM` is called, RPG copies the COMAREA status word into *VSTAT. If the status word is non-zero an error message is placed in the window and you are returned to the program with no abort. You must test *VSTAT and decide what to do. If an error has occurred, and you have specified closing the spoolfile after each print, the close is not done.

CLRMSG	Calls VPUTWINDOW. VSHOWFORM. VPUTWINDOW clears the window message buffer. If the RPG output buffer column 7 equals "I". VPUTWINDOW calls VSHOWFORM. After VSHOWFORM is called. RPG copies the COMAREA status word into *VSTAT. If the status word is non-zero. an appropriate message is placed in the window you are returned to the program with no abort. You must test *VSTAT and decide what to do.
CHMODE	Calls VTURNOFF. After VTURNOFF is called. RPG copies the COMAREA status word into *VSTAT. If the status word is non-zero. an appropriate message is placed into the window and the mode is not changed. You are returned to the program with no abort. You must check *VSTAT and decide what to do.
BLMODE	Calls VTURNON. Same as CHMODE.

Actions Not Modified

The actions that have not been modified are:

CHGNXT	Does not call any intrinsics. May modify COMAREA items NFNAME. REPEATAPP. FREEZAPP. depending on content of record.
GETNXT	Calls VGETNEXTFORM. If not in browse mode, VGETNEXTFORM calls VPUTBUFFER. blanking out the VPlus data buffer.
PUTMSG	Calls VPUTWINDOW. If the output buffer column 7 is not blank. the COMAREA item WINDOWENH is modified.
BADFLD	Calls VSETERROR. May modify COMAREA item WINDOWENH depending on the value in the output buffer column 14.
PUTDTA	Calls VPUTBUFFER.
INIT	Calls VINITFORM.
EDITS	Calls VEDITFIELDS.
NUMERR	No intrinsic calls. Sets the event code to 9. A following READ operation places the number of fields that failed a VPlus or user edit operation into the I-spec field allocated for this value.
GETDTA	No intrinsic calls. Sets the event code to 10. A following READ operation calls VGETBUFFER. No change from previous behavior for this call. either.
FINISH	Calls VFINISHFORM.

WRTBAT	Calls VWRITEBATCH. Sets the COMAREA item DELETEFLAG to False prior to the call. After VWRITEBATCH is called, if there is no error and it is not in browse mode, the COMAREA item RECNUM is incremented.
PREV	Calls VREADBATCH. If not in browse mode, sets the COMAREA item CMODE to "browse".
REREAD	Calls VREADBATCH.
NEXT	Calls VREADBATCH.
RESUME	No intrinsic calls. Sets the COMAREA item CMODE to "collect" and restores RECNUM and NFNAME to the values saved during a PREV action.
DELETE	Calls VWRITEBATCH. Sets the COMAREA item DELETEFLAG to True.
RDBTNU	Calls VREADBATCH. Sets the COMAREA item RECNUM to the record to be read.
GETFLD	No intrinsic calls. Sets the event code to 12. A following READ operation calls VGETFIELD. Again, no change from previous behavior for this call.
PUTFLD	Calls VPUTFIELD.
LOADFM	Calls VLOADFORMS.
UNLDFM	Calls VUNLOADFORM.

As mentioned previously, reading or changing data in the COMAREA is somewhat tricky in that it contains a mix of data, plus RPG does not have an internal binary data type. The method chosen to handle this is to declare the array *VC as an alphanumeric array of one character per element. You must read the data one character at a time. There are no conversion routines to make binary data readable (e.g., using DSPLY), so the best thing is to check the data using the COMP or TESTB operators. Alphanumeric data in the COMAREA (form names) may be displayed, one character at a time. To modify binary data, you must create the desired bit pattern yourself, using the BITOF/BITON operators. Admittedly this is not elegant, but it is the best RPG can do. Lets look at some examples.

Example 1

Suppose you wished to set the window enhancement to "B". This is alphanumeric, so it is easier. You want to modify the right byte of COMAREA word 8. This is *VC,16. The operation is:

```
        C                           MOVE "B"      *VC,16
```

To set the COMAREA length (word 3) to 60, note that the binary equivalent of 60 is "0000000000111100" (see any good computer science book for information on decimal to binary conversion). You need to clear and then set bytes 5 and 6. The operations are:

```
        C                           BITOF"01234567"*VC,5
        C                           BITOF"01234567"*VC,6
        C                           BITON"2345"    *VC,6
```

Note that `MOVE 0          *VC,5` or `Z-ADD0          *VC,5`, etc., does not work. `MOVE` places an ASCII "0" into the left byte of the "length" word, and `Z-ADD` yields a compile-time warning 6063 (result field must be numeric) and does not emit code for the operation.

Example 2

Suppose you wish to set retries to 8 instead of the default of 4. This is word 55 (bytes 109, 110) of the COMAREA. The byte bit pattern for this is "00001000". However, you now have another problem. The result field is only 6 characters long, but *VC,110 is 7 characters. The solution requires a new variable as follows:

```
        C                           Z-ADD109       X        30
        C                           BITOF"01234567"*VC,X
        C                           ADD  1         X
        C                           BITOF"01234567"*VC,X
        C                           BITON"4"       *VC,X
```

Now lets look at some coding examples.

Example 3

Suppose you are in a noisy environment and you would like to protect the SHOW action by retrying it five times with a 3-second pause between retries, in the event of a VSHOWFORM failure. You can do this as follows:

```
$CONTROL VPLUSCOM
H
                         .
                         .
C                    Z-ADDO         X       20
C                    Z-ADD3         TIME    40
C                    SETON                     80
C           AGAIN    TAG
C                    EXCPT
C           *VSTAT   COMP 0                    9191
C  N91               GOTO AOK
C                    SETOF                     91
C           X        IFEQ 5
C                    GOTO ABORT
C                    END
C                    ADD  1         X
C                    INTR PAUSE
C                    IPARM          TIME
C                    Z-ADDO         *VSTAT
C                    GOTO AGAIN
C           AOK      TAG
C                    SETOF                     80
C                      .
C                      .
C           ABORT    TAG
C                      .
C                      .
OTERMINALE        80
O                                   6 "SHOW  "
```

Example 4

Suppose you wish to do your own batch file operations. You wish to check the COMAREA status on a **VWRITEBATCH** call, and display an error message if it is non-zero.

Note If you have a form on your screen while executing the following code, it would mess it up.

The code for doing this is as follows:

```
$CONTROL VPLUSCOM
H
                                   .
                                   .
E                   BUFR        1 72
                                   .
                                   .
C                    Z-ADDO         *VSTAT
C                    INTR VWRITEBATCH
C                    IPARM          *VC
C           *VSTAT   COMP 0                    9191
C   91               EXSR WRBBAD
                                   .
                                   .
CSR         WRBBAD   BEGSR
C                    SETOF                     91
C                    Z-ADD72        BUFLEN  40
C                    Z-ADDO         ACTLEN  40      no. chars returned
C* do not clear the comarea status word.  we need it here.
C                    INTR VERRMSG
C                    IPARM          *VC
C                    IPARM          BUFR
C                    IPARM          BUFLEN
C                    IPARM          ACTLEN
C* display the message.
C           BUFR     DSPLY
C* pause 10 seconds to allow it to be read.
C                    Z-ADD10        TIME    40
C                    INTR PAUSE
C                    IPARM          TIME
C* reset status to zero.
C                    Z-ADDO         *VSTAT
C                    ENDSR
```

As you may see by now, once RPG/iX has opened the workstation, you can use the INTR/IPARM operators, *VSTAT, and *VC to write your VPlus applications using the VPlus intrinsics. If you use a mix of RPG actions and intrinsic calls, you must be aware that the actions in general do more than just call the intrinsics.

For example, if you do a SHOW action, RPG updates the COMAREA word 34 (SHOWCONTROL) bit 9 (PRE-LOAD) based on the value in column 7 of the O-spec record for SHOW (blank, "P", or invalid). This would modify anything you put in that spot prior to the action.

An annoying problem that currently exists in the RPG VPlus interface is that, if you wish to do an EDITS action on data in the VPlus buffer following an RDTERM action, you lose the ability to use the F1-F8 function keys. In other words, if you respond to RDTERM with F1-F8, do the EDITS, and then do a READ TERMINAL, the F1-F8 indicators do not get set, as you would expect.

Example 5

As a final example, the program shown below uses the new enhancement that overcomes this problem.

Note The forms file is not presented. If you wish to run this program, you would need to create your own forms file with the appropriate fields.

```
* This program shows how to use the new VPlus enhancement,
* both in accessing the COMAREA and in using INTR/IPARM oper-
* ators.  If you enter data and press ENTER, editing data as
* needed, the data will be copied to MPEFILE.  If you press f1-f7,
* no data is read.  Instead the form is re-displayed, and you may
* enter different data.  If you press f8, the program terminates.
* To do the EDITS operation, we now do as follows:
* We do a SHOW followed by a RDTERM action, and then use the new
* capability to call VFIELDEDITS.  We then access the COMAREA to
* see if f8 was the last key pressed, and if so we terminate the
* program.  If not, we check the 'numerrs' value, and if non-zero
* we try again, with an error message in the window.  If no errors
* are found, the program continues.
*
$CONTROL VPLUSCOM
H
F*****************************************************************
F*                File Specifications                            *
F*                                                               *
F*****************************************************************
FTERM    UD  V      110             WORKSTN   L3B
F                                                KFORMS VPFORMS
FMPEFILE O   F       72             DISC
E*****************************************************************
E*                Array Specifications                           *
```

```
E*                                                            *
E**************************************************************
E                    ERM     1   1 79
E                    MSG        79  1
I**************************************************************
I*                Input Specifications                        *
I*                                                            *
I**************************************************************
ITERM     AA  01    1 C0    2 C0    7 C1
I        OR         1 C1    2 C0    7 C1
I                                        1   20EVENT
I                                        3  17 FORM
I                                       18  210LENGTH
I                                       22  270DIGIT
I                                       28  33 ALPHA
I*
I* Define function key f1, f2, f3, f4, f5, f6, f7, f8.
I*
ITERM     BB  11    1 C0    2 C1
I        OR   11    1 C0    2 C2
I        OR   11    1 C0    2 C3
I        OR   11    1 C0    2 C4
I        OR   11    1 C0    2 C5
I        OR   11    1 C0    2 C6
I        OR   11    1 C0    2 C7
I        OR   18    1 C0    2 C8
C**************************************************************
C*                Calculation Specifications                  *
C*                                                            *
C**************************************************************
C                     BITOF"01234567"X       1
C          START      TAG
C                     SETOF                     0111
C                     SETOF                     18
C                     EXSR GETNXT
C                     EXSR SHOW
C                     EXSR EDITS
C   18                GOTO ENDPGM
C   11                GOTO START
C                     EXCPT          RECOUT
C                     GOTO START
C          ENDPGM     TAG
C                     SETON                     LR
C**************************************************************
C*                Subroutine GETNXT                           *
C*   INIT   = Initialize fields in current form               *
C**************************************************************
C          GETNXT     BEGSR
C                     MOVE "GETNXT"  ACTION  6
C                     EXCPT          ACTOUT
C                     MOVE "INIT  "  ACTION
```

```
C                    EXCPT          ACTOUT
C                    ENDSR
C*************************************************************
C*                 Subroutine SHOW                           *
C*   SHOW  = Display current form, initial data, any         *
C*           messages.                                       *
C*   RDTERM= Read input from terminal to data buffer.        *
C*************************************************************
C          SHOW      BEGSR
C  N90               MOVEAERM,1     MSG
C  N90               EXSR SHOMSG
C                    MOVE "SHOW  "  ACTION
C                    EXCPT          ACTOUT
C                    EXSR RDTERM
C                    ENDSR
C*************************************************************
C*                 Subroutine SHOMSG                         *
C*   DISPLAY MESSAGE, ANY NEW DATA.                          *
C*************************************************************
C          SHOMSG    BEGSR
C                    SETON                     54
C                    EXCPT
C                    SETOF                     54
C                    ENDSR
C*************************************************************
C*                 Subroutine EDITS                          *
C*    EDITS = Perform edits on fields in current form.       *
C*    FINISH= Perform final processing on current form.      *
C*    GETDTA= Write data in data buffer to user program.     *
C*************************************************************
C          EDITS     BEGSR
C                    SETOF                     90
C          AGAIN     TAG
C   11               SETOF                     11
C   18               GOTO ENDEDT
C   90               EXSR SHOW
C* this is where the new VPlus enhancement begins.
C* note: when testing the comarea "lastkey" and "numerrs" values,
C*       I have assumed the left byte of the 16-bit word is zero.
C*       In general, this will always be true (lastkey could be
C*       -1 for 3075/6 terminals, and numerrs would need more than
C*       256 fields on the screen to overflow into the left byte).
C                    INTR VFIELDEDITS
C                    IPARM          *VC
C* check comarea status. better be zero.
C          *VSTAT    COMP 0                    1818
C   18               GOTO ENDEDT
C* if comarea lastkey = 8, abort
C                    TESTB"4"       *VC,12         18
C   18               GOTO ENDEDT
C* else if lastkey  0, turn on indic. 11
```

```
C* note: if so desired, we could put in a specific test for
C*       each function key.
C           *VC,12    COMP X                   1111
C* if numerrs is not equal to 0, try again
C           *VC,14    COMP X                   9090
C   90                EXSR ERRMSG
C   90                GOTO AGAIN
C                     SETOF                    90
C                     MOVE "FINISH"  ACTION
C                     EXCPT          ACTOUT
C                     MOVE "GETDTA"  ACTION
C                     EXCPT          ACTOUT
C                     READ TERM                    H1
C           ENDEDT    TAG
C                     ENDSR
C*****************************************************************
C*               Subroutine RDTERM                               *
C*   RDTERM = Read input from terminal to data buffer.           *
C*****************************************************************
C           RDTERM    BEGSR
C                     MOVE "RDTERM"  ACTION
C                     EXCPT          ACTOUT
C                     ENDSR
C*   ERRMSG = put the error msg in the window for retry.
C           ERRMSG    BEGSR
C                     Z-ADD79        BUFLEN  40
C                     Z-ADD0         ACTLEN  40
C                     INTR VERRMSG
C                     IPARM          *VC
C                     IPARM          MSG
C                     IPARM          BUFLEN
C                     IPARM          ACTLEN
C*
C                     INTR VPUTWINDOW
C                     IPARM          *VC
C                     IPARM          MSG
C                     IPARM          ACTLEN
C                     ENDSR
O*****************************************************************
O*               Output Specifications                           *
O*                                                               *
O*****************************************************************
OTERM    E                ACTOUT
O                         ACTION     6
O        E        54
O                                    6 "SHOMSG"
O                                    8 "79"
O                                    9 "J"
O                         MSG       88
O*
OMPEFILE E                RECOUT
```

```
         O                                      6 "Digit:"
         O                         DIGIT       12
         O                                     20 "Alpha:"
         O                         ALPHA       26
         O                                     72 "VPFORMS"
**
              Test program for VPlus enhancement
         *
```

The above information is also available in the *Addendum to the HP RPG/iX Reference Manual* (30318-90016) and is available with the MPE/iX 5.5 Release.

FNUM Operator

Previously, the FNUM operator would only accept a numeric field for the result of the operation. This value was then kept as a packed decimal number, and had to be converted to binary (normally in an external subroutine) before it could be used in file intrinsics. This field can now be a two-character alphanumeric field where the file number is kept in binary format. This field can be used in an IPARM operation without further conversion when accessing intrinsics. For example, to determine the actual record length of a file named in F-specs (i.e., already opened by RPG), you could do:

```
     C                           FNUM filename  NUM      2
     C                           Z-ADD4         ITN04   40
     C                           INTR FFILEINFO
     C                           IPARM          NUM
     C                           IPARM          ITN04
     C                           IPARM          RECLEN  40
```

Note that NUM is not a DSPLYable item. If you want to see the file number, do another FNUM using a numeric field.

DSPLY Operator

Two new options, P and Q, have been added to the H-specification, column 48, for the DSPLY operator. These options allow factor1, the result field, and the user response to all appear on one line. It is your responsibility to adjust field lengths so that the fields and the reply for the result field all fit on the displayed line.

Option P displays DSPLY starting in column 1, followed by two blanks, followed by the factor 1 field, followed by two blanks, followed by the result field, followed by one blank (two if numeric; allows for the "-" sign), followed by the prompt for the new result field.

Option Q is the same as option P, except there is no DSPLY. Factor 1 begins in column 1.

VPlus Enhancements

by Kumar KN
Commercial Systems Division - India

Overview

VPlus version B.06.06, has been enhanced in the MPE/iX 5.5 Release. This version contains the following new functionality:

- Copying of processing specifications, which includes the following two copy commands:

`#COPYTO` *newfilename*	Copies the processing specifications defined for the field to *newfilename*.
`#COPYFROM` *oldfilename*	Copies the processing specifications in *oldfilename* into the processing specifications area of the field.

- Sensing the cursor position on a VPlus screen, which includes the following three new intrinsics:

`VARMSCP`	Arms or disarms cursor sensing capability.
`VGETSCPFIELD`	Returns information about the cursor position by field number on a VPlus screen.
`VGETSCPDATA`	Returns information about the cursor location by field number and row and column number on a VPlus screen.

Processing Specifications Copy Features

The processing specifications copy functionality provides the following features:

- Provides mass editing capability for forms.
- Lets users enter edit specifications once into a field (rather than retype them) and copy them out to other fields as required. This occurs typically when using form families.

This functionality is particularly useful when complex or lengthy processing specifications are involved.

Cursor Sensing Features

The cursor sensing functionality provides the following features:

- Provides a way to design screen interfaces based on user input.
- Allows Help menus to be displayed on the screen depending on whether a HELP field was accessed or not.
- Provides a useful way to duplicate other GUI functionality on a VPlus screen.

For more details on how to use these features, please refer to the "New Functionality in VPlus" article in the "Technical Articles" chapter of this *Communicator*.

HP Symbolic Debugger/iX Enhancement

by Sue Meloy
Support Technology Center

Overview

HP Symbolic Debugger/iX is a source-level symbolic debugger for C. Pascal. Fortran and Cobol. It provides the ability to single-step. display variables. set breakpoints. and perform other debugging functions at the source and assembly levels.

Support for Shared Global Data

As of MPE/iX-Express 3 based on General Release 5.0. HP Symbolic Debugger/iX version A.05.04 (HP31508). has been enhanced to support the ability to display and modify the values of global variables whose definitions are in an executable library (XL). If no debug information is available for a shared global variable. the debugger prints its address.

Before this support was available. the debugger would not print the correct values for shared global variables defined in XLs. In particular. it was unable to display the C `errno` variable. a vital error indicator. which is defined in XL.PUB.SYS.

HP Information Access Server SQL/iX Supports Newer Protocols

by Neil Alexander
Worldwide Response Center Operations

New Functionality

Beginning with MPE/iX-Express 2 based on General Release 5.0. HP Information Access Server SQL/iX (B3162A) has been enhanced to let PC users access HP 3000 servers using Microsoft WINSOCK, or HP WSOCKETS protocols.

These newer protocols run on HP Information Access for Windows, PC Client version A.06.02. If you need to upgrade your PC version, you can order the PC Client upgrade kit, product number B3139A through your HP Representative.

Benefits

- This enhancement provides improved capabilities to use third-party PC networking products.
- HP Information Access for Windows users can communicate with any of the HP Information Access servers using WINSOCK or HP WSOCKETS connections from their PC.
- Clients using LAN Manager can use protocols other than NetIPC.
- NetIPC network protocols will still be available to access HP 3000 servers.

Enhancements to HP Information Access Server MPE/iX

by Frank Heartney
Support Technology Center

The Information Access Server (B3160A) is the MPE/iX component of Information Access that returns TurboIMAGE, KSAM and MPE/iX flat file data from the HP 3000 to the PC. This article describes the enhancements to the Information Access Server beginning with MPE/iX-Express 3 based on General Release 5.0.

The enhancements described below are primarily of interest to Database Administrators, although the improved query optimization may be immediately visible to end-users.

Improvements in Access Server's Query Optimization

Access Server now does a better job of selecting an efficient order for reading the tables in multi-table queries. The change should be most noticeable for joins specified from the PC, as opposed to joins specified in view tables.

Unwanted Spool Files Eliminated

Access Server no longer leaves HDSPERR and HDSPNS spool files after a successful operation. These spool files are still present for aborts and other abnormal terminations, or if troubleshooting JCWs, like DPCOGEN, are active.

Support for IEEE Reals

Access Server now automatically recognizes IMAGE data items with the "E" data type, and accepts "E" as a data type for MPE/iX and KSAM files.

Improved Usage of MPE/iX File Characteristics

The maximum file size configurable in ADMIN has increased from 2,000,000 to 100,000,000 records in a file. At the same time, defaults for numextents and initialoc in FOPEN are used, so that files with large maximum sizes but just a few records, do not take up a lot of space.

Ability to Configure More Table Definitions

Up to 20000 tables definitions can be configured in the Access Server. The previous effective limit was 2079 table definitions.

Troubleshooting JCW

The output generated by the JCW DPCOGEN is now much more useful in describing how Access Server performs a query. To use the JCW, create a file called IAFILE in your logon group with the line:

```
SETJCW DPCOGEN 1
```

After executing a query from the PC, an HDSPERR spool file appears containing a summary of the order in which Access Server reads the source tables for the query, which keyed reads are used, and how multi-pass queries are split up.

ALLBASE/4GL Developer, Release B.07.00 Enhancements

by Lee Lequin
Worldwide Response Center Operations

Overview

This article outlines the new features and enhancements for ALLBASE/4GL Developer, release B.07.00, beginning with MPE/iX-Express 2 based on General Release 5.0.

HP ALLBASE/4GL is an advanced fourth-generation language. It enables you to design and implement application software by defining the required results, rather than the procedures necessary to achieve those results.

The HP ALLBASE/4GL system exists in two forms:

- HP ALLBASE/4GL Developer System (HP30601)

  The HP ALLBASE/4GL Developer System contains all the facilities of the run-time environment, plus an additional set of facilities to develop new applications or modify existing applications.
- HP ALLBASE/4GL Run-Time Environment (HP30602).

  The HP ALLBASE/4GL Run-Time Environment provides facilities for running HP ALLBASE/4GL applications and performing centralized system administration.

New Features

"New Features" Key

- Lets you preview new commands and improvements from the developer's main menu screen.
- Provides instructions for printing additional copies of the *Software Update Notice*

New LABEL Command

The new `LABEL` command allows developers to jump to a specific statement (instead of only a statement number) when using the `ENTER` and `SERIES` commands.

New "-V" (Verbose) Option for HP4STOA and HP4ATOS

The new "-V" (Verbose) option for HP4STOA and HP4ATOS monitors the progress of applications being saved and restored to the s-files.

Enhancements

Improvement to Application Definition Screen

Increases consistency in terminology with name change to filed label.

SQL Interface Enhancements

- Improves user efficiency when compiling, executing, syntax checking, and storing with pass-through mechanism for ALLBASE/SQL commands.
- Permits complex and nested queries for the `SELECT` command using the UNION operator.
- Expands supported commands to include `CLOSE`, `CONNECT`, `EXECUTE`, `IMMEDIATE`, `OPEN` and `RELEASE`.
- Allows `EXECUTE PROCEDURE` command from within ALLBASE/4GL to invoke SQL procedure.

New Option for FILE Command

Increases user capabilities with the new *REFETCH option in the `FILE` command. The `FILE *REFETCH` command revalidates data prior to carrying out an update to an SQL database.

Choice of External Editor

Permits user-selected editor when modifying functions, processes, SQL logic blocks, and SQL select lists.

Expansion of Item Limits

Increases the maximum sizes of the SQL select lists and SQL logic blocks.

Improvement to Menu Item Security

Retains menu item security when loading and unloading applications.

ALLBASE/SQL Enhancements

by The ALLBASE/SQL Team
Commercial Systems Division

ALLBASE/SQL version G.1 (B3892AA), available beginning with the MPE/iX-Express 3 based on Release 5.0, contains several major enhancements providing significant benefits in the following areas:

- ODBC PC API enhancements
- Standards
- High Availability
- Usability
- Tools
- Connectivity and client/server

If you are updating from an earlier release of ALLBASE/SQL, refer to the migration information in the "Special Considerations" section appearing later in this article. To order this product, refer to your HP Representative.

A list of ALLBASE/SQL reference manuals for MPE/iX is provided at the end of this article.

Important Details Please Read

The information in this article can only be found in this 5.5 release of the *Communicator*. Retain this copy if you will be using ALLBASE/SQL.

ODBC PC API Enhancements

In this release, the ALLBASE/PC API/ODBC product is bundled with ALLBASE/SQL, and the following enhancements are included:

- ODBCVIEW, the script that creates the ODBC catalog tables, has been enhanced to recognize TurboIMAGE keys, third-party indexes, and constraint indexes. A new view, ODBCADM.INDEXES, was created for this new index registration. Starting with the G.1 version of ODBC PC API, the ODBCVIEW script is bundled on the server side of ALLBASE/SQL. This enhancement eliminates the previous requirement for users to upload ODBCVIEW from the client to the server.
- A new flag has been added to ODBC PC API that allows users to turn off `AUTOCOMMIT` for ODBC client/server applications. `AUTOCOMMIT ON` is the default as defined by the ODBC standard, but some developers may choose to control logical transactions themselves.
- Support has been added in ODBC PC API for the following SQL statements: `BEGIN WORK`, `SET`, `SETOPT`, `TERMINATE`, `STOP`, and `RELEASE`.

- Support has been added for Microsoft's Wolverine 32-bit TCP/IP stack and WINSOCK driver.
- PC API allows users to enable data compression and, thereby, improve performance for data transfer in client/server applications.
- Binary Large Objects (BLOBS) are supported with the GUPTA PC API. Beginning with version A.G1, BLOBS support is a feature of ODBC PC API.
- Multi-Row Stored Procedures are a part of both the GUPTA and ODBC APIs. This feature allows users to execute multiple selects in a procedure and specify triggers and rules to validate data-entry input.

For more information on this subject, refer to the PC API ReadMe file and the "Connectivity and Client/Server" section appearing later in this article.

Standards

Enhancements that adhere to ANSI SQL standards are described in this section.

String Concatenation

String concatenation is allowed by using two vertical bars between strings.

High Availability

Enhancements related to high availability are described in this section.

SQLUtil STOREONLINE

The ability to use the SQLUtil `STOREONLINE` statement with the nonarchive logging option is now available. This enables you to access the DBEnvironment without concern for log file maintenance when a backup is in progress.

Usability

Enhancements related to ALLBASE/SQL usability are described in this section.

Default Number of Transactions

The default number of transactions for a newly created DBEnvironment has been expanded from 2 to 50.

WITH AUTOCOMMIT and FORCE Options for VALIDATE

The syntax of the `VALIDATE` statement has been enhanced in G.1 to add the `WITH AUTOCOMMIT` and `FORCE` options:

```
VALIDATE [FORCE          ]
         [DROP SETOPTINFO]
 ( MODULE { [Owner.]ModuleName }[, ... ]              )
 { PROCEDURE { [Owner.]ProcedureName }[, ... ]        }
 (     ( MODULES    )                                 )
 ( ALL {            } [WITH AUTOCOMMIT]               )
 (     ( PROCEDURES )                                 )
```

When the `WITH AUTOCOMMIT` clause is used, a `COMMIT WORK` statement is executed automatically after each `MODULE` or `PROCEDURE` is validated. This can reduce both log space and shared memory requirements for the `VALIDATE` command.

Example:

```
VALIDATE ALL MODULES WITH AUTOCOMMIT;
VALIDATE ALL PROCEDURES WITH AUTOCOMMIT;
```

When the `FORCE` clause is used, all sections associated with the `MODULE` or `PROCEDURE` are revalidated, regardless of whether they are valid or invalid.

Example:

```
VALIDATE FORCE ALL MODULES WITH AUTOCOMMIT;
VALIDATE FORCE ALL PROCEDURES WITH AUTOCOMMIT;
```

When the above statements are issued, every stored section in the database is forced to recompile using the latest release. These statements have essentially the same effect as preprocessing every program again that uses the database. We recommend that customers use the `FORCE` option when first installing a delta release of ALLBASE/SQL, e.g., when migrating from G.0 to G.1.

ALLBASE/SQL customers should be aware that changes to the internal data structure for queries have been made between G.0 and G.1. As a result, if G.1 is installed after a migration to G.0 has occurred, temporary performance degradation (due to the revalidation of these internal data structures) may be experienced. If the `VALIDATE` command is issued with the `FORCE` clause on all modules or procedures (as described above) during a period of low activity for the DBEnvironment, the Optimizer has current statistics on which to base its calculations. The effect of this action on performance degradation is minimal.

For more information on this subject, refer to the "Special Considerations" section appearing later in this article.

WITH AUTOCOMMIT Option for DELETE

The syntax of the `DELETE` statement has been enhanced in G.1 to add the `WITH AUTOCOMMIT` option.

$$\texttt{DELETE}\ \left[\texttt{WITH AUTOCOMMIT}\right]\texttt{FROM}\ \left\{\begin{array}{l}\left[\mathit{Owner}.\right]\mathit{TableName}\\ \left[\mathit{Owner}.\right]\mathit{ViewName}\end{array}\right\}$$

$$\left[\texttt{WHERE SearchCondition}\right]$$

When the WITH AUTOCOMMIT clause is *not* used, rows that qualify according to the SearchCondition are deleted internally in batches by ALLBASE/SQL.

When the WITH AUTOCOMMIT clause is used, a COMMIT WORK statement is executed automatically at the beginning of the DELETE statement and also after each batch of rows is deleted. This can reduce both log-space and shared-memory requirements for the DELETE statement. You cannot control the number of rows in each batch.

Example:

```
DELETE WITH AUTOCOMMIT FROM PurchDB.Orders
       WHERE OrderDate <"19830701";
```

Additional information:

1. The WITH AUTOCOMMIT clause cannot be used in these cases:
   a. When deleting rows from a TurboIMAGE data set.
   b. If a SET CONSTRAINTS DEFERRED statement is in effect.
   c. If a rule exists on the table and rules are enabled for the DBEnvironment. The DBA may wish to consider issuing a DISABLE RULES statement to temporarily disable rules for the DBEnvironment, issuing the DELETE WITH AUTOCOMMIT statement, and then issuing an ENABLE RULES statement to turn rule checking back on.
   d. In the DELETE WHERE CURRENT statement.
2. If an active transaction exists when the DELETE WITH AUTOCOMMIT is issued, then the existing transaction is committed.
3. When WITH AUTOCOMMIT is used, any previously issued SET DML ATOMICITY statements are ignored. For the duration of that DELETE command, row-level atomicity is used.
4. If the DELETE WITH AUTOCOMMIT statement fails, it may be true that some (but not all) rows that qualify have been deleted.
5. The DELETE WITH AUTOCOMMIT statement can be used in procedures, but a rule may not execute that procedure.

Decimal Operations

Maximum precision has been changed from 18 to 27.

Data type storage requirements for decimal are now:

```
DECIMAL (p[,s]) = 4 bytes (where p≤7) or
                  8 bytes (where 7<p≤15) or
                  12 bytes  (where 15<p≤23) or
                  16 bytes (where p>23)
```

The precision (p) and scale (s) of a DECIMAL result depends on the operation used to derive it. The following rules define the precision and scale that result from arithmetic operations of two decimal values having precision p_1 and p_2 and respective scales s_1 and s_2. Rules are also provided for the resulting precision and scale of aggregate functions that operate on a single expression having a precision of p_1 and a scale of s_1.

```
Addition and Subtraction
  p = MIN (27, MAX (p1 - s1, p2 - s2) + MAX(s1, s2)+1)
  s = MAX (s1, s2)

Multiplication
  p = MIN (27, p1 + p2)
  s = MIN (27, s1 + s2)

Division
  p = 27
  s = 27 - MIN (27, p1 - s1 + s2)

  Where p1 and s1 describe the numerator operand,
  and p2 and s2 describe the denominator operand.

MAX and MIN Functions
  p = p1
  s = s1

SUM Function
  p = 27
  s = s1

AVG Function
  p = 27
  s = 27 - p1 + s1
```

RENAME Table or Column

The **RENAME** statement defines a new name for an existing table or column in a DBEnvironment. Currently you can rename a table by dropping the old table then recreating it with the new name, loading the data, and recreating any indexes or other dependencies on the table. In this enhancement, the **RENAME SQL** command has been provided for renaming an existing table or column.

When using the **RENAME** command, data and grants made for tables are carried forward for the new name. All appropriate entries in the system catalog (e.g., all indexes, columns, default columns, constraints, referential authorization, rules, and user authorities), reflect the new name. All views dependent on a renamed table or column are dropped. If a table or a column has check constraints, then that table or column cannot be renamed. All synonyms for a renamed table or column return an error when used.

The **RENAME** command is not allowed for IMAGE/SQL tables.

SQL Syntax:

RENAME TABLE [*Owner.*] *TableName* TO *NewTableName*;
RENAME COLUMN [*Owner.*] *TableName.ColumnName* TO *NewColumnName*;

Parameters:

[*Owner.*]*TableName*	Designates the table to be renamed.
[*Owner.*]*TableName.ColumnName*	Designates the table column to be renamed.
NewTableName	New table name.
NewColumnName	New column name.

Tools

Tools enhancements to ALLBASE/SQL are discussed in this section.

POSIX Support

Starting with release G.1. The ALLBASE/SQL COBOL preprocessor (PSQLCOB) supports preprocessing and generation of Microfocus COBOL source code under POSIX. The preprocessor can be invoked from POSIX as follows:

```
PSQLCOB dbename -i progname -c MICROFOCUS
```

The database environment name is *dbename*. If this name is specified in a hierarchical fashion. the preprocessor converts it to an MPE/iX file specification. The following are both logically equivalent:

```
sampledb.cobout.sys
/SYS/COBOUT/SAMPLEDB
```

A relative hierarchical path cannot be used. e.g. `../sampledb`.

The *dbename* is not casesensitive. It should be noted that ALLBASE/SQL only allows the generation of DBEs as MPE/iX file specifications.

The name of the COBOL source file to be preprocessed is *progname*. This name is casesensitive and must be specified in a hierarchical fashion. Both relative and absolute pathnames may be used.

The preprocessor directive to generate Microfocus COBOL-compatible output files is `-c MICROFOCUS`. These files have suffixes such as .cbl and .sqlc.

Once a program has been preprocessed, it can be compiled and linked with the cob command. For example,

```
cob -xo abc abc.cbl
```

would compile and link the source module, abc.cbl, and create abc as the executable program file.

Connectivity and Client/Server

These are the connectivity and client/server enhancements to ALLBASE/SQL described in this section:

- Configuring the Listener
- Migration Issues for SERVER/CLIENT Using HP-UX 10.0 or Greater
- HPDADVR Run Queue Control
- ALLBASE/PC API/ODBC
- PC Communications Stacks Support

Also see the "ODBC PC API Enhancements" section in the beginning of this article.

Configuring the Listener

The listener, which runs on the server, provides three functions. It listens for TCP/IP connection requests; it validates new connection requests using the NETUsers file; and it sets up a direct communication link between the client and server. ARPA and NS (Network Services) TCP/IP listeners are both supported though ARPA is the most widely used.

As of the 5.0 release, for both ODBC and ALLBASE/NET, to configure the ARPA listener for TCP/IP access, do the following. First, verify that the HOSTS.NET.SYS and SERVICES.NET.SYS files exist on your HP 3000. If these files exist, verify that they contain the entries below. Make the necessary changes if required.

HOSTS.NET.SYS should contain the following line:

```
127.0.0.1     localhost loopback me myself local
```

SERVICES.NET.SYS should contain the following line:

```
DAServer    987/tcp       # SQL distributed access
```

If these files do not exist, perform the following to create them:

```
:COPY HOSTSAMP.NET.SYS;TO=HOSTS.NET.SYS
:COPY SERVSAMP.NET.SYS;TO=SERVICES.NET.SYS
```

The sample files contain the proper information and require no further editing.

To start the ARPA listener, logon as MANAGER.SYS, and type the following command:

```
ANSTART ARPA
```

It is recommended that this command be added to your system startup sequence such that the listener is started automatically after the network is started.

For more information on ALLBASE/NET, see the ALLBASE/NET User's Guide (P/N 36216-90031 for MPE/iX or P/N 36217-90093 for HP-UX). ALLBASE/NET is bundled with both IMAGE/SQL and ALLBASE/SQL.

Migration Issues for SERVER/CLIENT Using HP-UX 10.0 or Greater

Beginning with HP-UX 10.0, ARPA is the only TCP/IP interface for data communication through ALLBASE/NET. HP-UX supports other data communication types. However, ALLBASE/NET only supports ARPA. This affects all new and existing applications that use ALLBASE/NET.

Remote database access applications that specify NS as the data communication type do not work if the client and/or server machine is an HP 9000 S700 or S800 running HP-UX 10.0, or greater. The server node name entry must be changed from the NS node name to the ARPA host name. For the NETUsers file, the "client node name" must be changed from the NS node name to the ARPA host name. Two new commands, `MIGRATE ALIAS` and `MIGRATE USER`, were added to NETUtil to migrate the AliasDB and NETUsers files.

NETUtil. Changes described below must be made to NETUtil, since this is where "datacomm type" information is entered.

1. NETUtil commands that prompt you to specify a "datacomm type", e.g., `ADD ALIAS` and `CHANGE ALIAS`, issue error or warning messages when:
   a. The client and/or server machine is an HP 9000 S700 or S800 (the HP 9000 S300 and S400 are not supported on ALLBASE/SQL G.1)
   b. NS is entered as the "datacomm type".
2. Existing applications that run on an HP 9000 S700 or S800 client and/or server and use NS as the "datacomm type" must now use ARPA Services instead. This can be accomplished by making changes to the alias profiles in the AliasDB file on the client system and the user profiles in the NETUsers file on the server system. For the AliasDB file:
   a. "Datacomm type" must be changed from NS to ARPA.
   b. The "server node name" entry must be changed from the NS node name to the ARPA host name.
3. For the NETUsers file, the "client node name" must be changed from the NS node name to the ARPA host name.

HPDADVR Run Queue Control

HPDALSTN, a process running in the B queue, services remote database connections via ALLBASE/NET on MPE/iX by listening for and executing on incoming connection requests. When a connection request arrives from a peer process, the newly created process (HPDADVR.PUB.SYS) runs in either the C or D queues as follows:

Incoming Connection:	HPDADVR will run in the:
MPE/iX Session	C queue
MPE/iX Job	D queue
HP-UX	C queue

While these are still the defaults, it is now possible to run HPDADVR in the D queue for an MPE/iX Session or HP-UX Connection. Likewise, it is also possible to run HPDADVR in the C queue for an MPE/iX Job Connection. This can be accomplished by setting a new environment variable named `HPSQLJOBTYPE`. The values can be "D" or "C." When "D" is used, HPDADVR runs in the D queue. When "C" is used, HPDADVR runs in the C queue. HPSQLJOBTYPE must be set on the client requesting the remote connection.

For MPE/iX:

`:setvar HPSQLJOBTYPE "D"`	*Runs in D queue*
`:setvar HPSQLJOBTYPE "C"`	*Runs in C queue*

For HP-UX (C shell):

`setenv HPSQLJOBTYPE "D"`	*Runs in D queue*

ALLBASE/PC API/ODBC

The ALLBASE/PC API/ODBC product is now bundled with ALLBASE/SQL. HP PC API is an application programming interface that allows tools written with either the GUPTA interface or the ODBC interface to access ALLBASE/SQL and IMAGE/SQL on either an HP 3000 or HP 9000 server from a PC.

PC Communications Stacks Support

ALLBASE/PC API on client PCs allows connectivity to ALLBASE and IMAGE/SQL servers using a number of PC-based communication "stack" libraries.

PC LIBRARY	HP 3000 SERVER	HP 9000 SERVER
WINSOCK.DLL	FTP PC TCP/IP	FTP PC TCP/IP
	NetManage Chameleon	NetManage Chameleon
	Microsoft LanMan 2.2B[1]	Microsoft LanMan 2.2B[1]
	Walker Richer Quinn	Walker Richer Quinn
	Novell Lan Workplace[1]	Novell Lan Workplace[1]
	Wollongong Pathway	Wollongong Pathway
	Microsoft Wolverine (32bit)	Microsoft Wolverine (32bit)
WSOCKETS.DLL[2]	HP ARPA/NS 2.1, 2.2	HP ARPA/NS 2.1, 2.2
	Microsoft	Microsoft
	Walker Richer Quinn	Walker Richer Quinn
NetWare	NetWare 3.11, 4.21	NA
WLIBSOCL.DLL	Novell Lan Work Place 4.1	Novell Lan Work Place 4.1
	(with patch LWP 168)	(with patch LWP 168)

1 ALLBASE/PC API does not currently operate with this stack. This problem is being investigated.

2 Support for this library will be dropped when WINSOCK.DLL can adequately replace it.

Note Stack libraries are not supplied with ALLBASE/PC API. They can be obtained from their manufacturers.

Special Considerations

ALLBASE/SQL is auto-installable. However, if you are updating from an earlier release of ALLBASE/SQL, you must perform the ALLBASE/SQL migration to migrate your DBEnvironments to the G.1 format.

If You Have Version G.0

If your release of ALLBASE/SQL is G.0, execute the SQLINSTL script to migrate to version G.1. ALLBASE/SQL has added new views and modified some existing views to support TurboIMAGE indexes in IMAGE/SQL. The SQLINSTL script is provided in the G.1 version of ALLBASE/SQL to make it easy for a DBA to move to a delta release, such as G.1. Using SQLINSTL ensures that you have access to the most recent version of the SYSTEM and CATALOG views, and it also uses `VALIDATE' FORCE` statements to revalidate all stored sections.

If you do not execute SQLINSTL, stored sections are revalidated when they are accessed by end-users of the database. Exclusive locks are obtained on system catalog tables during revalidation.

Note Hewlett-Packard recommends that you use SQLINSTL when you install the delta release to avoid concurrency problems that may arise if revalidation occurs during production hours.

Examples:

```
MPE/iX  isql.pub.sys
        isql=> connect to 'mydbe';
        isql=> start sqlinstl.pub.sys;
        isql=> connect to 'mydbe';
        isql=> exit;

HP-UX   /usr/bin/isql
        isql=> connect to 'mydbe';
        isql=> start /usr/lib/allbase/hpsql/sqlinstl;
        isql=> connect to 'mydbe';
        isql=> exit;
```

Please read the SQLINSTL file on your system for more information.

Customers who are using `ARCHIVE MODE LOGGING` must make a backup of the DBEnvironment after using SQLINSTL. This backup must be used if rollforward recovery is to be performed at some point in the future. Customers installing G.1 cannot apply rollforward recovery to a backup created using the G.0 version (or earlier) of ALLBASE/SQL.

If You Have Version E.1 or F.0

If your release of ALLBASE/SQL is E.1 or F.0. use SQLMIG to migrate to version G.1. A backup of the DBE should be done prior to running. The steps listed below also appear in the *ALLBASE/SQL Database Administration Guide* (36216-90005).

Use the following procedure to convert a DBEnvironment from either an E.1 or an F.0 format to the G.1 format:

1. Prior to updating the operating system and ALLBASE/SQL software, do the following for each DBEnvironment that will be migrated:
    a. Run ISQL.PUB.SYS and issue a `START DBE` command. This ensures that the DBEnvironment is logically consistent in the event that it has not been accessed since a system failure occurred.
    b. Run SQLUTIL.PUB.SYS and issue the `STORE` command to backup each DBEnvironment.

Note Log files are not stored using this command. In addition, you should use the `SHOWDBE` command to ensure that all parameters are OK.

2. Backup the ALLBASE/SQL software (system backup will suffice).

3. Update the operating system and the ALLBASE/SQL software.
4. Enter the command:

```
    :RUN SQLMIG.PUB.SYS
```

5. For each DBE that is to be migrated, check for potential errors during the migration by using the PREVIEW command, which follows:

```
    SQLMIGRATE=> PREVIEW 'DBEnvironmentName' FORWARD;
```

Note

Since PREVIEW is not a read-only command, you should make sure that you have a backup of the DBEnvironment prior to issuing the PREVIEW command.

During the PREVIEW check, you may receive messages stating that there is not enough space in the SYSTEM DBEFileSet. If this occurs, use the following commands to create a new DBEFile and add it to the SYSTEM DBEFileSet:

```
SQLMIGRATE=> CREATE DBEFILE DBEFileName
    WITH PAGES = DBEFileSize, NAME = 'SystemFileName';

SQLMIGRATE=> ADD DBEFILE DBEFileName TO DBEFILESET SYSTEM;
```

Note that the syntax of these commands is the same as in ISQL. Repeat this step until no errors are encountered and SQLMigrate returns the following message:

```
    The proposed migration should be successful
```

6. Issue the MIGRATE command as follows:

```
    MIGRATE=> MIGRATE 'DBEnvironmentName' FORWARD;
```

When the forward migration has successfully completed, SQLMIG purges the old log files and performs a START DBE NEWLOG to create a new log file using the parameters stored in the DBECON file.

Example:

```
START DBE NEWLOG BEGINNING (MON, JUL 19, 1993, 4:12 PM)

START DBE 'DBENAME' NEWLOG
   BUFFER = (100,24),
   TRANSACTION = 50,
   MAXIMUM TIMEOUT = 3600 SECONDS,
   DEFAULT TIMEOUT = 30 SECONDS,
   RUN BLOCK = 37

   LOG DBEFILE log1 WITH PAGES = 250,
      NAME = 'DBELog1';

START DBE NEWLOG SUCCEEDED (MON, JUL 19, 1993, 4:13 PM)
```

The DBEnvironment is ready to be accessed. If you desire archive-mode logging, you must run SQLUtil and issue the `STOREONLINE` command.

7. Exit SQLMIG as follows:

   ```
   SQLMIGRATE=> EXIT;
   ```

8. If the `START DBE NEWLOG` (issued by SQLMIG) should fail for any reason, you must run ISQL and issue the `START DBE NEWLOG` command from ISQL.
9. Run SQLUtil and issue a `SHOWDBE` command to check the parameters of the new version of the DBEnvironment. If you wish to use archive-mode logging, run SQLUtil and use the `STOREONLINE` command. Issue the `SHOWLOG` command to verify that `ARCHIVE MODE` is set properly.

   You can then exit SQLUtil. At this point the DBE should be ready for access.

If You Have a Version Prior to E.1

If your release of ALLBASE/SQL is earlier than E.1, you must first update to ALLBASE/SQL, release E.1 or F.0, then perform the ALLBASE/SQL migration update to release G.1.

Additional Information

For additional information, please refer to the following ALLBASE/SQL reference materials:

Up and Running with ALLBASE/SQL (36389-90011)

ALLBASE/SQL Reference Manual (36216-90001)

ISQL Reference Manual for ALLBASE/SQL and IMAGE/SQL (36216-90096)

HP ALLBASE/QUERY User's Guide (92534-90011)

ALLBASE/SQL Database Administration Guide (36216-90005)

ALLBASE/SQL Message Manual (36216-90009)

ALLBASE/SQL Advanced Application Programming Guide (36216-90100)

ALLBASE/NET Users Guide (36216-90031)

ALLBASE/SQL Performance and Monitoring Guidelines (36216-90102)

HP PC API Users Guide for ALLBASE/SQL and IMAGE/SQL (36216-90104)

IMAGE/SQL with TurboIMAGE/XL Enhancements

by Bharati V. Desai
Commercial Systems Division

In the MPE/iX 5.5 Release, IMAGE/SQL version B.G1.06 (HP36385) has been enhanced significantly. This article discusses the new enhancements in MPE/iX Release 5.5, which incorporates the enhancements made in MPE/iX-Express 3 as well.

Important Details Please Read

The information in this article can only be found in this 5.5 release of the *Communicator*. Retain this copy if you will be using IMAGE/SQL.

What is IMAGE/SQL?

IMAGE/SQL is the next generation of TurboIMAGE/XL, which allows existing TurboIMAGE customers on MPE/iX to evolve to relational technology while preserving their existing application investment in TurboIMAGE/XL. It is an enhanced combination of TurboIMAGE/XL and what was formerly known as ALLBASE/TurboCONNECT (ATC). IMAGE/SQL provides relational access to your TurboIMAGE/XL data using the industry-standard Structured Query Language (SQL). While ATC only provided read capability, HP IMAGE/SQL includes both read and write capability using SQL ANSI standard functionality.

Closely tuned to the architecture of Hewlett-Packard computers, IMAGE/SQL gives you flexibility in designing and using new SQL database applications. At the same time, you can continue to use your existing TurboIMAGE/XL applications without noticeable differences.

Note

With IMAGE/SQL, you do not have to convert, recompile, or make changes to your existing applications.

Your TurboIMAGE/XL database can now be simultaneously accessed by SQL applications as well as your existing TurboIMAGE/XL applications. With IMAGE/SQL, you can continue to enhance your existing TurboIMAGE/XL applications or develop new ones.

IMAGE/SQL includes the client/server components HP ALLBASE/PC API and ALLBASE/NET. This means that PC-based client applications using industry-standard SQL can use IMAGE/SQL to access TurboIMAGE/XL data on an HP 3000 host system. With the increased usage of PCs and rapid adoption of client/server technology, HP IMAGE/SQL extends the availability of your TurboIMAGE/XL data to report writers, 4GLs, and other applications running on a PC network.

As an IMAGE/SQL user, you benefit by:

- Leveraging your investment

  Your existing investment in TurboIMAGE/XL is preserved while you take advantage of relational technology.

- Accessing new tools

  Available SQL client/server tools allow you to view data from TurboIMAGE/XL, HP ALLBASE/SQL, or third-party databases.

- Increasing your productivity

  Because SQL uses a common interface to access TurboIMAGE/XL, ALLBASE/SQL, and third-party SQL databases, your productivity will improve.

Components of IMAGE/SQL

IMAGE/SQL includes TurboIMAGE/XL, a restricted version of ALLBASE/SQL, and a database administration tool called IMAGESQL (formerly known as ATCUTIL), that links them together. The following discussion describes the components of IMAGE/SQL in more detail.

TurboIMAGE/XL	A set of programs and procedures used to create and maintain TurboIMAGE/XL databases. This constitutes the former TurboIMAGE/XL product prior to bundling it with IMAGE/SQL.
ALLBASE/SQL	A set of programs and procedures used to create, maintain, and access relational database environments (also known as DBEnvironments). The SQL part of IMAGE/SQL is ALLBASE/SQL, with a restriction on the amount of data you can store in user SQL tables. IMAGE/SQL includes components needed to provide relational access to TurboIMAGE/XL databases, including the following:

- ISQL lets you enter SQL statements at the keyboard and observe query results, messages, and other information on a video display. ISQL is the main tool for IMAGE/SQL programmers and database administrators to create and modify DBEnvironments, load and unload data, and test SQL statements.
- SQLUtil is a database administrators tool for maintaining relational database environments.
- SQLGEN is a database administrators tool used to generate commands to recreate all or part of an existing database environment.
- SQLMON is an online diagnostic tool used to monitor the activity of a database environment.

- Preprocessors are tools used to convert C, COBOL, FORTRAN, or Pascal source programs containing SQL statements into source code that can be compiled.
- ALLBASE/SQL PC API is software that provides access from a Microsoft Windows based PC to ALLBASE/SQL and TurboIMAGE/XL databases residing on HP-PA RISC database servers. You need to use one of several available PC client/server or decision support tools to build your application or query.
- ALLBASE/NET works in conjunction with ALLBASE/SQL PC API to connect users with databases residing on an HP 3000 server.

IMAGESQL (ATCUTIL)

A database administrator's tool used to manage TurboIMAGE/XL databases in a relational-access environment. IMAGESQL registers information about a TurboIMAGE/XL database into the system catalog of an ALLBASE/SQL DBEnvironment. In IMAGE/SQL, the TurboIMAGE/XL data sets look like SQL tables, with one exception. SQL statements that retrieve, update, insert, or delete data ultimately get routed to the TurboIMAGE/XL storage manager when TurboIMAGE/XL data sets are referenced.

As a result, the large suite of client/server tools that support ALLBASE/SQL now support TurboIMAGE/XL as well.

In IMAGE/SQL, the size of your TurboIMAGE/XL databases is not restricted in any way, but you can only store 12 MegaBytes (3000 4K pages) of data in user-defined SQL tables. This is because you get a restricted version of ALLBASE/SQL with IMAGE/SQL. For an unrestricted copy of ALLBASE/SQL, you must purchase ALLBASE/SQL.

Information Access Server

This is both the HP 3000-based server and a component of the Hewlett-Packard Information Access SQL/iX. This is bundled with IMAGE/SQL.

How Do You Get IMAGE/SQL?

IMAGE/SQL can now be purchased by TurboIMAGE/XL users. You must have TurboIMAGE/XL on your support contract to be eligible to purchase IMAGE/SQL.

With IMAGE/SQL, you automatically receive a restrictive copy of ALLBASE/SQL. This restrictive copy allows you to create up to 3000 4K (1 K=1024 bytes) pages of SQL table data. For an unrestricted copy of ALLBASE/SQL, you must purchase ALLBASE/SQL.

Additional Information

IMAGE/SQL can be automatically installed using HP AUTOINST. The manuals included with IMAGE/SQL are:

- *HP IMAGE/SQL Administration Guide* (36385-90001)
- *Getting Started With HP IMAGE/SQL* (36385-90008)
- *HP PC API Users Guide for ALLBASE/SQL and IMAGE/SQL* (36216-90104)
- *ALLBASE/SQL Message Manual* (36216-90009)
- *ISQL Reference Manual for ALLBASE/SQL and IMAGE/SQL* (36216-90096)

Updating IMAGE/SQL

If you are updating from an earlier release of IMAGE/SQL and you have databases which are already ATTACHed to one or more DBEs. you must DETACH and ATTACH again to benefit from the performance enhancements. If you do not, you may experience significant performance degradation.

Also, If you have created SQL data using an earlier release of IMAGE/SQL and are now updating to a latter release, you must perform the ALLBASE/SQL migration to migrate your DBEnvironment to the current G.1 format. For more information, refer to the Special Considerations section in the ALLBASE/SQL Enhancements article in this chapter or the ALLBASE/SQL Database Administration Guide (36216-90005).

IMAGE/SQL Enhancements

Detailed information regarding enhancements to IMAGE/SQL is described in this section.

Indexed Access in IMAGE/SQL

The ALLBASE/SQL Optimizer is now aware of TurboIMAGE/XL search items, key items, and third-party indexes (Bradmark's Superdex and DISC's Omnidex). This includes both P (packed decimal) and Z (decimal) data types. The Optimizer decides which indexes to use and the proper order of operations to ensure that the most efficient path is used. As a result, data is retrieved more efficiently when a mapped column represents a TurboIMAGE/XL search item, key item, or a third-party index. Chained reads

(DBFIND and DBGET) or calculated reads (DBGET mode 7) instead of serial scans are used whenever possible. In addition, similar performance gain is also evident for deletes as well as updates.

The version of third-party software that supports indexed access in IMAGE/SQL is also required from both third parties.

Important Details Please Read

Third-party index functionality is limited to partial generic key access and range retrieval only. The Optimizer derives an index scan based on the least number of page I/Os for the data set. Hence, it may select an index on the column not being used in the SQL statement for the data set.

In order to make the indexed access enhancement, modifications were made to IMAGE/SQL, IMAGESQL (ATCUTIL), TurboIMAGE, DBUTIL, ALLBASE/SQL, as well as the third-party software. Changes to IMAGESQL, DBUTIL, and new views added to ALLBASE/SQL system catalog are briefly discussed here as you must use them for this enhancement.

IMAGESQL (ATCUTIL)

The IMAGE/SQL utility, also referred to as IMAGESQL, was formerly called ATCUTIL. The IMAGESQL commands `ATTACH`, `DETACH`, `SPLIT`, and `UPDATE` have been modified as follows:

- **ATTACH**
  - TurboIMAGE/XL master keys, detail search items:

    `ATTACH` triggers registration of all TurboIMAGE/XL master keys and detail search items as indexes in the DBE specified by the `SET SQLDBE` statement. All master keys, except P and Z data types, are registered as unique indexes. All detail search items as well as master P and Z key types are registered as non-unique indexes. If your database is already attached, you need to do `DETACH` and `ATTACH`. The indexes are registered only in the specified DBE.
  - Third-party keys:

    If your database is configured for third-party indexes (TPI) and `enabled` for indexing, it attempts to register the third-party indexes as non-unique B-tree indexes in all DBEs to which the database is attached. In other words, if your database is enabled for TPI, you do not need to issue the `DETACH` command. You may use a temporary DBE initially to attach to it, which causes third-party indexes to be registered in all DBEs to which the database is attached. This means that with a single `ATTACH` (to a new or temporary DBE), you can register third-party indexes in all DBEs to which the database is attached.

Note

Registering in several DBEs is a complex task. When this design issue was discussed with users. two schools of thought emerged: TurboIMAGE/XL users not using the third-party indexes voted for one DBE. while third-party index users proposed that all DBEs should be handled. It was decided to register TurboIMAGE/XL keys and search items in the specified DBE and third-party indexes in all DBEs to which the database enabled for TPI is attached. For third-party indexes, this is consistent with DBUTIL's **DISABLE/ENABLE INDEXING** commands which only apply to third-party indexes.

Please note that the TurboIMAGE/XL compound item and items split using IMAGE/SQL, as well as the third-party keyword indexes are not supported in this release of the indexing enhancement. That is, if the key/search item is a compound item or is split using the IMAGESQL utility. the SQL optimizer will not choose an index scan for it. Similarly, for third-party keyword and composite indexes, there may not be a performance gain.

Example of ATTACH:

The following example illustrates how you may get the SQL Optimizer to recognize your indexes using an ATTACH command. The database "testdb" is already configured for TPI; however. it is not attached to any DBE.

```
:RUN IMAGESQL.PUB.SYS


HP36385B B.G1.08 IMAGE/SQL Utility THU, JAN 2, 1996, 5:18 PM
COPYRIGHT HEWLETT-PACKARD COMPANY 1993

>>turbo testdb
>>dbe maindbe
DBE does not exist, do you want to create one?  [Y/N]:  Y
Creating DBE now ...
>>attach
Split 3 compound source field(s) (ATCWARN 32062).
Mapped 74 source table/source field name(s) (ATCWARN 32062)
>>exit
:
```

TurboIMAGE/XL keys/search items. as well as the third-party indexes. are now registered in the ALLBASE/SQL SYSTEM CATALOG.

- **DETACH**

  DETACH only affects the DBE named in the SET SQLDBE statement. Information about all tables, indexes, etc. is removed from the specified DBE. It works the same way for TPI as well as for a non-TPI database.

- **SPLIT**

  SPLIT drops the table, that is, all information about the table including the indexes, is removed from the system catalog. Information about the table, TurboIMAGE/XL keys/search items, and third-party indexes (if any) are added again to the ALLBASE/SQL SYSTEM CATALOG.

  This command only affects the DBE named in the SET SQLDBE statement. If you want a similar change in other DBEs, use the SET DBE command for another DBE and use the same SPLIT command. Note that if you split a TurboIMAGE/XL master key or a detail search item, it is not registered as an index. Also, if you split an item and it has a third-party index on one of the components other than the first one, you may not see the performance improvement.

- **UPDATE TYPE**

  This is similar to SPLIT, however, it may deal with one or more tables as the UPDATE command is applicable to either one table or all tables in the database that have the specific type. The UPDATE TYPE command only affects the DBE named in the SET SQLDBE statement.

- **DBUTIL (for Indexed Access in IMAGE/SQL)**

  DBUTIL is enhanced for third-party indexes. The ENABLE, DISABLE, and SHOW statements have been modified for third-party index registration (not TurboIMAGE/XL keys/search items registration) as follows:

  - **ENABLE database FOR INDEXING**

    If your database is already attached to at least one DBE and you enable it for indexing (third-party indexes must be configured before you use the ENABLE command), an attempt is made to register the third-party indexes in all DBEs to which the database is attached. If the registration fails in one DBE, it proceeds to register in subsequent existing DBEs. If such an attempt fails in one or more DBEs, upon completion of the last attempt, a message is displayed as follows:

    ```
    Registration of Third-Party Indexes failed
        from these HP SQL DBEnvironments
    ```

    The names of DBEs in which the third-party registration failed follows the message.

- **DISABLE database FOR INDEXING**

  If the third-party indexes are registered in the ALLBASE/SQL SYSTEM CATALOG, the DISABLE database for INDEXING command triggers removal of the third-party indexes information from all DBEs to which the database is attached. The removal of third-party index information from all DBEs is necessary, otherwise, results can be unpredictable at run-time. If removal of this information fails in one DBE, it proceeds with subsequent existing DBEs. If such an attempt fails in one or more DBEs, upon completion of the last attempt, a message is displayed as follows:

  ```
  Dropping of Third-Party Indexes failed
      from these HP SQL DBEnvironments
  ```

  This is followed by names of DBEs in which the dropping of third-party indexes failed.

  In this case, the database still remains ATTACHed, however, the ALLBASE/SQL Optimizer has no knowledge of the third-party indexes as that information is removed. This command does not impact TurboIMAGE keys/search items, which remain known to the Optimizer.

  In brief, DETACH (of IMAGESQL) only deletes information about third-party indexes from one DBE. DISABLE (of DBUTIL) attempts to remove information on third-party indexes from all DBEs to which the database is attached as the database is turned off for TPI.

- **SHOW database FLAGS or SHOW database ALL**

  If the third-party indexes are registered in one or more DBEs, the FLAGS option displays a message indicating that the third-party indexes are registered. The ALL option displays all DBEs to which the database is attached, as well as the DBEs in which the third-party indexes are registered.

New Views

Four new views have been added to the SQL system catalog to enable the DBA and other users to view TurboIMAGE/XL indexes as well as third-party indexes known to the SQL Optimizer. The Optimizer determines the most efficient path to the desired data, such as which indexes to use and the proper order of operations. The DBA should periodically issue an UPDATE STATISTICS command on each TurboIMAGE/XL data set to refresh the statistics that are displayed on these views and ensure that the Optimizer makes correct decisions. After all statistics have been updated, the DBA should issue VALIDATE commands to revalidate any stored sections that may exist on the data sets.

For example:

```
isql=> update statistics for table music.albums;
isql=> update statistics for table music.composers;
isql=> validate all modules with autocommit;
isql=> validate all procedures with autocommit;
```

For more information, please see the Maintenance chapter of the *ALLBASE/SQL Database Administration Guide* (35216-90005).

The four views are described as follows:

- **SYSTEM.IMAGEKEY and CATALOG.IMAGEKEY**

  The DBA can view all TurboIMAGE/XL keys associated with a database by examining SYSTEM.IMAGEKEY. Other users can view the TurboIMAGE/XL keys to which they have access by examining CATALOG.IMAGEKEY.

  For example, the DBA can issue:

```
isql => select * from system.imagekey;


----------------------------------------------------
INDEXNAME          | TABLENAME    | OWNER | UNIQUE | ...
----------------------------------------------------
ALBUMCODE_M1       | ALBUMS       | MUSIC |  1     | ...
COMPOSERNAME_M1    | COMPOSERS    | MUSIC |  1     | ...
SELECTIONNAME_A1   | SELECTIONS_A| MUSIC |  1     | ...
ALBUMCODE_D1       | SELECTIONS   | MUSIC |  0     | ...
SELECTIONNAME_D2   | SELECTIONS   | MUSIC |  0     | ...
COMPOSERNAME_D3    | SELECTIONS   | MUSIC |  0     | ...
ALBUMCODE_D1       | LOG          | MUSIC |  0     | ...
SELECTIONNAME_D2   | LOG          | MUSIC |  0     | ...
```

The following columns exist in both SYSTEM.IMAGEKEY and CATALOG.IMAGEKEY (PRIMARIES, SCCCOUNT, and DCCCOUNT are not calculated or used in G.1):

INDEXNAME	Name of the TurboIMAGE/XL item, plus a suffix. The following suffixes are used by IMAGESQL: _M1 is used when registering manual master keys. _A1 is used when registering automatic master keys. _D*n* is used when registering search keys on detail data sets (where *n* can be 1 to 16, depending on the path number of keys.)

TABLENAME	Name of the TurboIMAGE/XL data set on which the key is defined.
OWNER	Name of the TurboIMAGE/XL database (or the owner name used during the `ATTACH`) on which the key is defined.
UNIQUE	Uniqueness indicator: 0 if duplicates are allowed, i.e., the index is not unique. 1 if duplicates are not allowed. i.e., the key is unique. Keys on master data sets (both automatic and manual) are always unique, except keys defined on P and Z (decimal) data types. Search items on detail data sets are always non-unique.
NUMC	Number of columns in the index. NUMC is always 1.
COLNUMS	A vector of 16 SYSTEM.COLUMN entries, which identifies the column numbers that make up the key. In ISQL, each column number is displayed as a field of 4 hexadecimal digits.
NDISTINCT	Number of distinct key values.
PRIMARIES	Number of primary slots used.
SCCCOUNT	Synonym Chain Cluster Count, which is a measure of how well the data is clustered on pages as the synonym chain (also known as the secondary chain) is traversed.
DCCCOUNT	Detail Chain Cluster Count, which is a measure of how well the data is clustered on pages as the detail chain is traversed.

■ SYSTEM.TPINDEX and CATALOG.TPINDEX

The DBA can view all third-party indexes (TPIs) associated with a database by examining SYSTEM.TPINDEX. Other users can view the TPIs to which they have access by examining CATALOG.TPINDEX. For example, the DBA can issue:

```
sql => select * from system.tpindex;


-----------------------------------------------
INDEXNAME     | TABLENAME   | OWNER | UNIQUE | ...
-----------------------------------------------
ACODE_T1      | ALBUMS      | MUSIC |  0     | ...
ALBUMC_T2     | SELECTIONS  | MUSIC |  0     | ...
SELECTNAME_T3| SELECTIONS  | MUSIC |  0     | ...
COMPOSER_T4   | COMPOSERS   | MUSIC |  0     | ...
```

The following columns exist in both SYSTEM.TPINDEX and CATALOG.TPINDEX (NPAGES, NLEVELS, NLEAVES, NDISTINCT, NFIRST, NPERKEY, and CCOUNT are not calculated or used in G.1):

INDEXNAME	Name of the third-party index. The following suffix is used by IMAGESQL when registering third-party indexes: _T*n* (Where *n* can be 1 to 400 depending on the TPIs that exist on the database.)
TABLENAME	Name of the data set on which the TPI is defined.
OWNER	Name of the TurboIMAGE/XL database (or the owner name used during the ATTACH) on which the TPI is defined.
UNIQUE	Uniqueness indicator: 0 if duplicates are allowed, i.e., the index is not unique. 1 if duplicates are not allowed, i.e., the index is unique.
CLUSTER	Clustering indicator: 0 if the index is not a clustering index. 1 if the index is a clustering index. TPIs are always registered as non-clustering.
NUMC	Number of columns in the index.
COLNUMS	A vector of 16 SYSTEM.COLUMN entries, which identifies the column numbers that make up the index. In ISQL, each column number is displayed as a field of 4 hexadecimal digits.
NPAGES	Number of pages containing the index.

NLEVELS	Number of levels in the B-tree index.
NLEAVES	Number of leaf pages in the B-tree index.
NDISTINCT	Number of distinct key values.
NFIRST	Number of distinct first key values.
NPERKEY	Number of pages per key.
CCOUNT	Cluster count, which indicates how well the data of the index are sorted: 0 before first UPDATE STATISTICS statement is processed. *n* (efficiency of clustering) best clustering if *n*=NPAGES of table indexed; worst if *n*=NROWS of table indexed.
CTIME	Time of creation: yyyymmddhhsstt.
COLDIRS	A vector of 16 direction entries, which indicates the direction of the corresponding column in the index definition. In ISQL, each column number is displayed as a field of 4 hexadecimal digits. 5 Ascending. 6 Descending.

Security Enhancements

- **ADD USER and UPDATE USER**

  The `ADD USER` and `UPDATE USER` commands in the IMAGESQL utility have been enhanced to allow the user class in addition to password. This method allows any reserved word and/or special character including a semicolon to be used as the TurboIMAGE/XL password. Password remains as an optional parameter for backward compatibility.

  New Syntax:

```
AD[D] USER user[@account]WITH { CLASS=classnum }
                              { PASS=password  }
[,MODE=modenum]

                               { CLASS=classnum }
U[PDATE] USER user[@account]TO { PASS=password  } [,...]
                               { MODE=modenum   }
```

- **DISPLAY USER**

  The `DISPLAY USER` command has been enhanced to display the `OWNER NAME` as in the `DISPLAY MAP` command. Also, it has been enhanced to allow a user who is not the database creator to show user information without displaying passwords. Passwords are displayed only for the database creator (DBC).

TurboIMAGE/XL K8 Data Types as SQL Date/Time Data Types

The `UPDATE TYPE` command of IMAGE/SQL version B.G1.07 or later in MPE/iX Release 5.5 has been enhanced to allow you to update your `K8` data types of TurboIMAGE/XL to SQL data types `DATE`, `TIME`, `DATETIME`, or `INTERVAL`. The default `K8` data type mapping to SQL is `CHAR[16]`. By doing this, dates, times, and intervals can be entered through the SQL interface and stored directly in the TurboIMAGE/XL database in the internal format of the ALLBASE/SQL date/time data types. This is particularly useful in a PC client/server environment where many of the PC tools have date and time data types which map directly to the ALLBASE/SQL date/time data types through the PC API layers supported with IMAGE/SQL and ALLBASE/SQL.

$$\texttt{U[PDATE] TYPE} \left\{ \begin{array}{l} \textit{sourcetype}\ \texttt{IN} \left\{ \begin{array}{l} * \\ \textit{mappedtbl} \end{array} \right\} \\ \texttt{IN}\ \textit{mappedtbl.mappedcol} \end{array} \right\} [\texttt{TO}\ \textit{newtype}]$$

where *newtype* can also be *DATE*, *TIME*, *DATETIME*, or *INTERVAL*

```
Some examples are:

UPDATE TYPE IN table1.K8item1    TO DATE
UPDATE TYPE IN table2.K8item2    TO TIME
UPDATE TYPE IN table3.K8item3    TO DATETIME
UPDATE TYPE IN table4.K8item4    TO INTERVAL
UPDATE TYPE K8 IN *              TO DATE
```

With this added functionality, however, you may want to read from and write to these K8 data type fields in your TurboIMAGE/XL database using TurboIMAGE/XL intrinsics as well. Since data in these fields is stored in the internal ALLBASE/SQL date/time format, a way of converting it into a readable character string is necessary. Similarly, it is necessary to convert character data into the ALLBASE/SQL date/time format before it can be inserted into your TurboIMAGE/XL database using the TurboIMAGE/XL intrinsics.

To handle this conversion, new externally callable procedures are provided. The collection of these new procedures are called "DATE/TIME Application Programming Interface", that is, "DATE/TIME API" in abbreviated form. The DATE/TIME API is possible in ALLBASE/SQL version `A.G1.14` and IMAGE/SQL version `B.G1.10`. This section describes the DATE/TIME API and how to use it. For additional information on ALLBASE/SQL date/time functions, refer to the "Date/Time Functions" discussion in the "Expressions" chapter of the *ALLBASE/SQL Reference Manual* (36216-90001).

There is a one-to-one mapping between the Date/Time API routines and the ALLBASE/SQL date/time functions as below:

```
 Date/Time          ALLBASE/SQL
API routine     Date/Time function
-----------     ------------------

 DBTODATE           TO_DATE
 DBTOTIME           TO_TIME
 DBTODTTM           TO_DATETIME
 DBTOITVL           TO_INTERVAL
 DBTOCHAR           TO_CHAR
 DBTOINT            TO_INTEGER
```

The functionality of the API routine is equivalent to its counterpart ALLBASE/SQL Date/Time function.

The descriptions of all the parameters for each of the Date/Time API routines are as follows.

- **DBTODATE**

Syntax:

DBTODATE (*charval*, *stringlen*, *format*, *fmtlen*, *dateval*, *error*)

Parameters:

charval	4-byte address of an array of characters holding the character representation of a date.
stringlen	4-byte integer length of *charval* in bytes.
format	4-byte address of an array of characters holding the format specification of the input character string. *charval*. Only valid format specifications for the TO_DATE function are allowed.
fmtlen	4-byte integer length of *format* in bytes.
dateval	4-byte integer address of a 16-byte buffer in which the resulting date in ALLBASE/SQL internal format is stored.
error	4-byte address of a 4-byte integer where error code is returned. *error* is set to 0 if no error occurred. Otherwise, it is set to the DBERR code for the error returned.

- **DBTOTIME**

Syntax:

`DBTOTIME` (*charval, stringlen, format, fmtlen, timeval, error*)

Parameters:

charval	4-byte address of an array of characters holding the character representation of time.
stringlen	4-byte integer length of *charval* in bytes.
format	4-byte address of an array of characters holding the format specification of the input character string, *charval*. Only valid format specifications for the TO_TIME function are allowed.
fmtlen	4-byte integer length of *format* in bytes.
timeval	4-byte integer address of a 16-byte buffer in which the resulting time in ALLBASE/SQL internal format is stored.
error	4-byte address of a 4-byte integer where error code is returned. *error* is set to 0 if no error occurred. Otherwise, it is set to the DBERR code for the error returned.

- **DBTODTTM**

Syntax:

`DBTODTTM` (*charval, stringlen, format, fmtlen, dttmal, error*)

Parameters:

charval	4-byte address of an array of characters holding the character representation of date-time value.
stringlen	4-byte integer length of *charval* in bytes.
format	4-byte address of an array of characters holding the format specification of the input character string, *charval*. Only valid format specifications for the TO_DATETIME function are allowed.
fmtlen	4-byte integer length of *format* in bytes.
dttmal	4-byte integer address of a 16-byte buffer in which the resulting time in ALLBASE/SQL internal format is stored.
error	4-byte address of a 4-byte integer where error code is returned. *error* is set to 0 if no error occurred. Otherwise, it is set to the DBERR code for the error returned.

- **DBTOITVL**

  Syntax:

  `DBTOITVL` (*charval, stringlen, format, fmtlen, itvlval, error*)

  Parameters:

charval	4-byte address of an array of characters holding the character representation of interval value.
stringlen	4-byte integer length of *charval* in bytes.
format	4-byte address of an array of characters holding the format specification of the input character string. *charval*. Only valid format specifications for the TO_INTERVAL function are allowed.
fmtlen	4-byte integer length of *format* in bytes.
itvlval	4-byte integer address of a 16-byte buffer in which the resulting interval in ALLBASE/SQL internal format is stored.
error	4-byte address of a 4-byte integer where error code is returned. *error* is set to 0 if no error occurred. Otherwise, it is set to the DBERR code for the error returned.

- **DBTOCHAR**

  Syntax:

  `DBTOCHAR`(*dateval,datatype,format,fmtlen,charval,bufflen,error*)

  Parameters:

dateval	4-byte address of the 16-byte date, time, datetime, or interval value stored in the ALLBASE/SQL date/time format to be converted.
datatype	4-byte integer representing the data type of the input, DATEVAL. It must be one of the following values:

10	Date
11	Time
12	DateTime
13	Interval

format	4-byte address of an array of characters holding the format specification of the desired format for the character string result. Only valid format specifications for the TO_CHAR function are allowed.
fmtlen	4-byte integer length of *format* in bytes.

charval	4-byte address of a character buffer to put result. This routine will fill this buffer with the character string representation of the date/time value, blank-filling to the end of the buffer as indicated by the length. If the character string representation for the date/time value is longer than the specified length of the buffer, the character string will be truncated to specified length.
bufflen	4-byte integer length of *charval* buffer in bytes.
error	4-byte address of a 4-byte integer where error code is returned. *error* is set to 0 if no error occurred. Otherwise, it is set to the DBERR code for the error returned.

■ DBTOINT

Syntax:

DBTOINT(*dateval, datatype, format, fmtlen, intval, error*)

Parameters:

dateval — 4-byte address of the 16-byte date, time, datetime, or interval value stored in the ALLBASE/SQL date/time format to be converted.

datatype — 4-byte integer representing the data type of the input, DATEVAL. It must be one of the following values:

10	Date
11	Time
12	DateTime
13	Interval

format — 4-byte address of an array of characters holding the format specification specifying which component (month, day, hour, etc.) of the input, *dateval*, should be converted to the integer. Only valid format specifications for the TO_INTEGER function are allowed.

fmtlen — 4-byte integer length of *format* in bytes.

intval — 4-byte address of a 4-byte buffer where the integer result gets stored.

error	4-byte address of a 4-byte integer where error code is returned. *error* is set to 0 if no error occurred. Otherwise, it is set to the DBERR code for the error returned.

Other Enhancements

- **Predicate Level Locking**

  To promote a high level of concurrency, predicate level locking, instead of set level, is used whenever possible. Only records containing qualified criteria are locked.

- **Packed Decimal**

  IMAGE/SQL has been enhanced to support IMAGE packed decimal from P16 to P28 data types.

- **TABLE as Database and Data Set Name**

  IMAGE/SQL has been enhanced to allow TABLE as the database name for the `SET TURBODB` command and as a data set name for the DISPLAY MAP command.

- **Multi-Connect**

  The Multi-Connect feature (available in ALLBASE/SQL) is now also available in IMAGE/SQL. This feature enables you to simultaneously connect to more than one database environment.

- **String Concatenation**

  String concatenation is allowed by using the operator, '||' between strings in an expression.

```
SELECT Partnumber || VendPartNumber, UnitPrice
from PurchDB.SupplyPrice;
```

TurboIMAGE/XL Enhancements

The following are the enhancements to TurboIMAGE/XL, a major component of IMAGE/SQL. Most of these enhancements were first released in MPE/iX-Express 3 based on General Release 5.0, except Dynamic Multiple Database Transaction and DBRECOV enhancements, which are released in MPE/iX Release 5.5. Many of these enhancements were requested by TurboIMAGE/XL users and you can use them regardless of your database being attached to an `SQL DBE`.

Jumbo Data Sets

This enhancement allows IMAGE users to create data sets greater than 4GBytes in size. A data set of this type, called a Jumbo data set, can span more than one MPE file. The naming convention uses POSIX extensions.

For example, a data set named SALES03 with four multiple files results in a total of five files with the following filecodes and names:

```
SALES03          filecode -408
SALES03.001      filecode -409
SALES03.002      filecode -409
SALES03.003      filecode -409
SALES03.004      filecode -409
```

SALES03 is the Chunk Control file. It has information about the other files, but no user data of its own. The remaining files in the data set are called Chunk Data files or chunks.

The impact of this enhancement on various components of TurboIMAGE/XL is described below:

DBSCHEMA

To specify a Jumbo data set, the `$CONTROL JUMBO` statement must be included in the schema before any data sets. Then any data set whose capacity is greater than 4GB automatically becomes a Jumbo data set. If the `$CONTROL JUMBO` command is not specified, an error is generated for data sets exceeding the 4GB limit. The `$CONTROL NOJUMBO` command can be used for turning off the Jumbo feature in DBSCHEMA.

Important Details Please Read

Third-party and diagnostic tools **must be** enhanced to handle Jumbo data sets. Otherwise, customers using Jumbo data sets will not be able to use the current third-party and diagnostic tools. To have a jumbo data set for your existing database(s), use one of the available third-party tools which support this feature.

DBUTIL

Some commands of DBUTIL have been enhanced for Jumbo data set enhancement, however, there are no changes in the externals. The `SHOW` command has been enhanced to indicate the presence of Jumbo data sets in a database. The `CREATE` command has been enhanced to create the required chunk control and chunk data files. The `MOVE` command now enables you to move either the chunk control file or a specific chunk data file to a different device.

DBSTORE

`DBSTORE` does not store a Jumbo data set. Instead, the `STORE` command must be used for this purpose. Also, POSIX names must be specified. For example,

```
STORE ./SALES@
```

TurboIMAGE/XL Intrinsics

There is no impact on the externals of any TurboIMAGE/XL intrinsics other than new error codes generated by the Jumbo data set enhancement. There are two new DBINFO modes, 206 and 207. Mode 206 (short format) gives the number of chunks in a data set. Mode 207 (long format) identifies the size of each chunk in terms of IMAGE records in addition to providing the number of chunks.

DBINFO Buffer Layout

The following is the DBINFO buffer layout for the new modes:

DBINFO mode 206

Element	Contents
1	# of chunks in a Jumbo data set

If the data set is not a Jumbo data set, zeroes are returned for the number of chunks.

DBINFO mode 207

Element	Contents
1	# of chunks in a Jumbo data set
2	0
3-4	Size of chunk 1 (# entries. not # of blocks!)
5-6	Size of chunk 2 (# entries. not # of blocks!)
7-8	...
2n + 3	Size of chunk n (# entries. not # of blocks!)

Total size: (n + 1) * 4 bytes.

If the data set is not a Jumbo data set, then zeros are returned for the number of chunks.

Backward Compatibility

Compatibility problems can only occur when Jumbo data sets are created. When Jumbo data sets exist, versions of TurboIMAGE/XL prior to C.06.00 are not able to open the database. The IMAGE database version, contained in the root file for databases with Jumbo data sets, is C03, whereas the version is C02 without Jumbo data sets.

Jumbo Data Sets and Dynamic Data Set Expansion

At the present time. Jumbo data sets do not work with Dynamic Data Set Expansion. DBSCHEMA has a check to prevent this. The requirement for these two to work together is recognized by HP.

Limits

The number of chunks is currently limited to 99. However, users are not likely to reach this limit unless chunks are very small. The TurboIMAGE limit for the maximum block number per file is the main limiting factor for the number of chunks, as this limits the size of a Jumbo data set to approximately 40GB, or about 10 chunks. Under normal circumstances, each Chunk Data file, except the last one, is about 4GB.

New error message

```
    -1200  Erase of a jumbo data set failed.
```

Dynamic Roll-Back of Multiple Database Transaction

TurboIMAGE/XL version C.06.10 (or later) in MPE/iX Release 5.5 has been enhanced to allow dynamic roll-back of a multiple database transaction. This is an extension of two existing features—dynamic roll-back, which permits only one database per dynamic transaction, and the multiple database transaction, commonly known as MDBX, which is static in nature and used with user logging. Unlike static MDBX, logging and DBRECOV are not needed with dynamic transactions, because the database can be recovered dynamically. Nevertheless, you may use user logging and DBRECOV if you so desire. In this article, the term MDBX is used to refer to a static multiple database transaction and DMDBX for a dynamic multiple database transaction. The dynamic transaction is bracketed by DBXBEGIN and DBXEND intrinsics and the static transaction is bracketed by DBBEGIN and DBEND intrinsics.

As in a dynamic transaction for one database prior to the version C.06.10, you can now define a DMDBX by employing DBXBEGIN, DBXEND and DBXUNDO intrinsics. DBXBEGIN designates the beginning of a sequence of TurboIMAGE/XL procedure calls that are to be regarded as a dynamic transaction for the purposes of logging and dynamic roll-back recovery. Similarly, DBXEND designates the end of a sequence of TurboIMAGE/XL procedure calls regarded as a dynamic transaction. DBXUNDO is used to dynamically roll-back a sequence of TurboIMAGE/XL procedure calls that completed successfully until that point inside a dynamic transaction. For DMDBX, the underlying concept is the same as that in the present dynamic transaction feature for a single database except that you can now include more than one database in your base parameter and use a new mode 3 to designate a DMDBX. The new syntax for the base parameter now accommodates multiple base IDs as in a static MDBX transaction. Furthermore, you must use the same base parameter used for DBXBEGIN in the corresponding DBXEND/DBXUNDO intrinsics and a new mode 3. The base parameter contains a transaction ID returned by TurboIMAGE/XL in the first two bytes, and is used for the corresponding DBXEND/DBXUNDO intrinsics.

The primary benefit of DMDBX is that you can dynamically roll-back updates to multiple databases, which is not possible either in a static MDBX or a dynamic transaction prior to this enhancement. Another advantage of DMDBX is that you need not use user logging to employ dynamic roll-back functionality. However, unlike MDBX, in order to use DMDBX, it is required that a DBCONTROL mode 7 be done for every database you want to include in a DMDBX, after DBOPEN of that database and before using it in the DBXBEGIN intrinsic. DBCONTROL mode 7 enables the database for deadlock detection, which when encountered, returns an error 26 instead of triggering a process hang.

If the calling process is logging, DBXBEGIN, DBXEND, and DBXUNDO cause a log record to be written to the log file to identify the beginning, end, and roll-back, respectively, of a dynamic transaction. For DMDBX, logging should be either **disabled** or **enabled** for all databases involved in the DMDBX. In addition, if logging is enabled, the same logid needs to be used for all databases as well. In case of DMDBX when logging is enabled, multiple log records, one for each database, will be written to the log file.

In order to take the advantage of DMDBX, you need to modify your application to use the enhanced intrinsics as given below.

- **DBXBEGIN**

Syntax:

$$\texttt{DBXBEGIN} \;,\left\{\begin{matrix} \textit{base} \\ \textit{baseidlist} \end{matrix}\right\} \;\textit{text,mode,status,textlen}$$

Parameters:

base	Name of the array used as the base parameter when opening the database. The first element of the array must contain the base ID returned by DBOPEN.
baseidlist	Name of the integer array containing the base IDs of the databases which are involved in a DMDBX. Use *baseidlist* when calling DBXBEGIN mode 3 (DMDBX). The layout of this array is shown here (note that each element is a halfword, or two bytes):

Element	**Contents**
1-2	Application program must set these two halfwords to binary 0s before calling DBXBEGIN. After returning to the calling program, these two halfwords contain the transaction ID. Use this same *baseidlist* with

	the corresponding DBXEND or DBXUNDO intrinsics.
3	Number of base IDs involved in the DMDBX. This must be a number between 1 and 15 inclusive.
4-*n*	Base IDs of the databases involved in the DMDBX. Each base ID occupies one half-word or 2 bytes and it is the first halfword of the *base* parameter used to call TurboIMAGE/XL intrinsics.

text
: Name of an array up to 256 halfwords long that contains user ASCII or binary data to be written to the log file as part of the DBXBEGIN log record. The *text* argument is used to assign each particular transaction a distinct name.

mode
: Integer indicating the type of transaction desired as follows:

Mode 1:	Indicates a dynamic transaction which spans only one database.
Mode 3:	Indicates a dynamic transaction spanning multiple databases. If user logging is enabled for the databases, mode 3 generates *multiple* entries in the log file, one for each database.

status
: Name of an array of 10 halfwords in which TurboIMAGE/XL returns status information.

textlen
: Integer equal to the number of halfwords to be logged from the *text* parameter, or is a negative integer equal to the number of bytes. Length can be zero.

Note

1. DBXBEGIN is not allowed with DBOPEN mode 2.
2. DBXBEGIN is not allowed with AUTODEFER enabled.
3. DBXBEGIN cannot be called if another transaction started by DBXBEGIN or DBBEGIN is active.
4. DBCONTROL mode 7 needs to be called before calling DBXBEGIN mode 3.

New error messages added are:

```
-139   Invalid Number of base IDs.
-140   Bad base ID list.
```

▪ DBXEND

Syntax:

DBXEND ,$\left\{\begin{matrix} \mathit{base} \\ \mathit{baseidlist} \end{matrix}\right\}$ *text,mode,status,textlen*

Parameters:

base	Name of the array used as the base parameter when opening the database. The first element of the array must contain the base ID returned by DBOPEN.
baseidlist	Name of the integer array containing the base IDs of the databases which are involved in the DMDBX. Use the same *baseidlist* used with DBXBEGIN when calling DBXEND mode 3 to end the DMDBX. The layout of this array is same as in DBXBEGIN. except that the transaction ID is already set in the first two halfwords by DBXBEGIN.
text	Array of up to 256 halfwords long that contains user ASCII or binary data to be written to the log file as part of the DBXEND log record.
mode	Must be a halfword equal to 1. 2 or 3.
	Mode 1: End of dynamic transaction spanning one database.
	Mode 2: End of dynamic transaction spanning one database. started with DBXBEGIN mode 1. and write contents of the transaction management (XM) logging buffer in memory to disk. If logging is enabled. the contents of the logging buffer in memory will also be written to disk.
	Mode 3: Indicates the end of a DMDBX started with DBXBEGIN mode 3. If user logging is enabled for the databases. mode 3 generates *multiple* entries in the log file. one for each database in the DMDBX. in order to mark the end of a dynamic transaction.
status	Name of an array of 10 halfwords in which TurboIMAGE/XL returns status information about the procedure.

textlen	Integer equal to the number of halfwords to be logged from the *text* parameter, or is a negative integer equal to the number of bytes. Length can be zero.

DBXEND returns an error condition if it is called without a prior matching call to DBXBEGIN. DBXEND is not necessary after a DBXUNDO.

■ DBXUNDO

Syntax:

$$\texttt{DBXUNDO}\ ,\left\{\begin{array}{l} base \\ baseidlist \end{array}\right\} ,text,mode,status,textlen$$

Parameters:

base	Name of the array used as the base parameter when opening the database. The first element of the array must contain the base ID returned by DBOPEN.
baseidlist	Name of the integer array containing the base IDs of the databases which are involved in the DMDBX. Use the same *baseidlist* parameter of DBXBEGIN mode 3. when calling DBXUNDO to roll-back a DMDBX.
text	Array of up to 256 halfwords long that contains user ASCII or binary data to be written to the log file as part of the DBXUNDO log record.
mode	Must be a halfword equal to 1 when employing *base* as the base parameter or 3 when using *baseidlist* with the matching DBXBEGIN call.

Mode 1:	Dynamically roll-back DBPUT, DBDELETE, and DBUPDATE intrinsics which completed successfully since the matching DBXBEGIN mode 1 call.
Mode 3:	Dynamically roll-back DBPUT, DBDELETE, and DBUPDATE intrinsics which completed successfully inside the DMDBX to their respective databases since the matching DBXBEGIN mode 3 call. If user logging is enabled for the databases, mode 3 generates *multiple* entries in the log file, one for each database, in order to mark the roll-back of the dynamic transaction.

status	Name of an array of 10 halfwords in which TurboIMAGE/XL returns status information about the procedure.
textlen	Integer equal to the number of halfwords to be logged from the *text* parameter, or is a negative integer equal to the number of bytes. Length can be zero.

Caution After DBXUNDO is called, the current record pointer, current path, current list, and chronological order prior to the call to DBXBEGIN may not be restored.

DBXUNDO cannot be called to roll-back a transaction started by DBBEGIN. DBXUNDO returns an error condition if it is called without a prior matching call to DBXBEGIN. DBXEND is not necessary after DBXUNDO. DBXUNDO rolls back the entire transaction, and then transactions continue according to the logic of the program. In general, DBXUNDO or DBXEND must be the last intrinsic of a dynamic transaction to be executed. It designates the end of that dynamic transaction.

■ DBCONTROL

Syntax:

```
DBCONTROL  ,base,qualifier,mode,status
```

Parameters:

base	Name of the array used as the base parameter when opening the database. The first element of the array must contain the base ID returned by DBOPEN.
qualifier	Currently ignored by DBCONTROL.
mode	Must be a halfword equal to 1, 2, 5, 6, 7, 9, or 10.

Mode 1: Turn on the deferred output option. If AUTODEFER has not been enabled for the database (using DBUTIL's ENABLE command), mode 1 enables the deferred output option for the duration of only the current DBOPEN. When the database is closed, AUTODEFER will no longer be enabled. Mode 1 is not allowed while a dynamic transaction is active.

Mode 2: Turn off the deferred output option. If AUTODEFER has been enabled for the database (using DBUTIL's ENABLE command), mode 2 disables the deferred output option for the duration of only the current DBOPEN. When the database is closed, AUTODEFER will again be enabled.

Mode 5: Enable the critical item update option. If CIUPDATE has been ALLOWED for the database (using DBUTIL's SET command), mode 5 enables the option for the current DBOPEN until either a DBCONTROL mode 6 call disables the option or the database is closed. You can call mode 5 successfully if the CIUPDATE setting for the database equals ON, but the call has no impact on the option setting for the current process unless an intervening call to DBCONTROL mode 6 disabled the option. If the CIUPDATE flag is DISALLOWED, a call to mode 5 returns an error, -82. The CIUPDATE option is available only in database access modes 1, 3, and 4.

Mode 6: Disable the critical item update option. If CIUPDATE has been set to ON for the database (using DBUTIL's SET command), mode 6 disables the option for the current DBOPEN until either a DBCONTROL mode 5 call enables the option or the database is closed. If the CIUPDATE option setting for the database equals ALLOWED and the process has called DBCONTROL with mode 5 to enable the option, mode 6 disables the option for that same process. The CIUPDATE option is available only in database access modes 1, 3, and 4.

Mode 7: Allow the database to be included in the DMDBX. DBCONTROL mode 7 needs to be done once, for every database before including it in DBXBEGIN mode 3 call for DMDBX. Mode 7 is used programmatically to allow it for DMDBX and remains activated until the database is closed or the application terminates.

Another use of mode 7 is to activate the database for deadlock detection. In case of a deadlock, DBLOCK will return an error, 26 (`Imminent deadlock`) instead of causing a process hang.

Mode 9: Enable the HWMPUT option of DBPUT for the current DBOPEN. This causes DBPUT to try placing entries at the high-water mark first instead of using the delete chain head first.

Mode 10: Disable the HWMPUT option of DBPUT for the current DBOPEN. This causes DBPUT to try placing entries at the delete chain head first. This is the default action if DBCONTROL is not called.

status	Name of an array of 10 halfwords in which TurboIMAGE/XL returns status information about the procedure.

For more information on DBXBEGIN, DBXEND, DBXUNDO, and DBCONTROL, refer to the TurboIMAGE/XL manual.

DBRECOV

Beginning with MPE/iX 5.0 Express 3, DBRECOV has been enhanced to support significantly larger logging cycles. Internal tables have been expanded by four to five times the size of the internal tables in the previous versions of DBRECOV.

As of MPE/iX Release 5.5, DBRECOV version C.06.10 has been enhanced to perform rollforward recovery of TurboIMAGE/XL databases stored using the new TurboSTORE/iX 7x24 True-Online Backup (described as *7x24 true-online backup* with the `ONLINE=START` or `ONLINE=END` options).

Benefits. The benefits of using 7x24 true-online backup to store TurboIMAGE/XL databases are as follows:

- You can store one or more TurboIMAGE/XL databases, along with their related files such as the TC file and the third-party index files, if any.
- You can actually perform 7x24 true-online backup, that is, the database can be stored when it is open for `read/write access`, or `closed`.
- You don't need to stop the user log process before backup, if logging is enabled. Also, you don't need to start a new log cycle after the backup. Therefore, the rollforward recovery, if enabled for rollforward, will commence from the middle of the log file, and it

is all transparent to you. That is, you don't need to use any new option for DBRECOV!

Note If your database is opened exclusively, such as with mode 3 or 7, 7x24 true-online backup does not store your database, and reports an error:

```
DATABASE NOT STORED: UNABLE TO STORE SOME DATABASE FILES
BASED ON THE SELECTION CRITERIA, NO FILES WERE STORED. (S/R 1713)
```

To summarize, unlike DBSTORE and STORE, you may have your database(s) open for access, and you can still perform the backup, and later restore it to perform rollforward recovery, when necessary.

Databases Enabled for Roll-Back Recovery. Databases enabled for roll-back recovery are not affected by using either DBSTORE, STORE, or 7x24 true-online backup. Furthermore, if your database is enabled for rollforward recovery and you don't use 7x24 true-online backup to store your database, you are not impacted by this enhancement to DBRECOV, that is, you may continue to exercise your existing procedures.

If you don't use DBRECOV to perform recovery, or your database is not enabled for user logging, you can still use 7x24 true-online backup to store your database(s). For more information on 7x24 true-online backup, refer to the following documents:

- *Communicator* article, "Introducing the TurboSTORE/iX 7x24 True-Online Backup Product," in Chapter 3, "System Management"
- *Communicator* article, "STORE and TurboSTORE/iX 7x24 True-Online Backup New Functionality," in Chapter 10, "Technical Articles"
- *STORE and TurboSTORE/iX Products Manual* (B5151-90001)

Database Rollforward Recovery Using 7x24 True-Online Backup. This article emphasizes how to perform rollforward recovery when your database is stored using 7x24 true-online backup.

When performing a 7x24 true-online backup, you want to know at what point in time the backup occurred, especially when your database is open for modification. This is helpful when performing a rollforward recovery so you know at what point all files were in a logically consistent state at the same time for backup and later a subsequent restore for recovery. This is the point called the *sync point*, where all data set files are synchronized. Also, it means this the point when the TurboIMAGE/XL database is quiesced for a short duration, that is, it is in a logically consistent state and there is no active transaction in progress. All ongoing transactions, if any, are allowed to be completed before the *sync point*.

The TurboSTORE 7x24 True-Online Backup product has introduced two new options to be used with the **STORE** command. The **ONLINE=START** option allows you to have the *sync point* at the beginning of a 7x24 true-online backup, and the **ONLINE=END** option permits you to have the *sync point* at the end of a 7x24 true-online backup. The **ONLINE=START** option has the following advantages over **ONLINE=END**:

- Allows your database to be restored on an earlier version of MPE/iX.
- Allows faster partial (selective, not @.@.@) restores, since sync at end requires RESTORE to read the log files (used for backup) from the end of the last piece of media.
- Spreads log data throughout the backup media, and hence, is less vulnerable to media errors.

Important Details Please Read

All true-online backups created with the sync point at the beginning can be restored on any MPE/iX system. However, backups created with the sync point at the end can only be restored or verified on MPE/iX Release 5.5 or later. If you know at the time of performing the backup that the database(s) must be restored onto an earlier system, create the backup with the sync point at the beginning. This is independent of using user logging and DBRECOV.

To perform rollforward recovery of your database stored using 7x24 true-online backup, the following requirements must be adhered to:

- Your system must be up on MPE/iX Release 5.5 or later.
- **LOGID** must include the **AUTO** option. If it was not used in the **GETLOG** command, use the **ALTLOG** command to include the **AUTO** option.

  ```
  :GETLOG imageid;log=image001,DISC;pass=wontell;auto
  ```

  or

  ```
  :ALTLOG noautoid;auto
  ```

  Refer to the *MPE/iX Commands Reference Manual* (Volumes I and II) (32650-60115) for more information on these commands.
- **LOG** process must remain active when storing the database even when the database is not open. That is, **LOG logid,STOP** must not be issued before storing the database. This is because the logging information which is dynamic in nature, is incorporated in the root file when the database is stored. This dynamic logging information can only be obtained when the log process is active. It is used later when performing rollforward recovery.

- One of the new options, ONLINE=START or ONLINE=END of the TurboSTORE ONLINE command, must be employed when performing the backup.

```
:FILE musicbk;dev=tape
:STORE music;*musicbk;online=start
```

- After the backup is completed, purge all log files preceding the one in use when the backup was initiated. It is essential that you retain the one used when backup was initiated and the ones following that. These are the log files that will be needed later to perform rollforward recovery.

  To find out the log file used by 7x24 true-online backup, employ the SHOW database ALL command of DBUTIL, which is enhanced to display the time, date, and the log file name used for the 7x24 true-online backup. Some examples of the SHOW database ALL command when using 7x24 true-online backup are given below.

  1. When 7x24 true-online backup is used, but user logging is not used:

```
Access is enabled.
   .
   .
Logging is disabled.
   .
   .
Database last stored using True-Online Backup on
   THU, JAN 18, 1996,  3:49 PM.
```

  2. When 7x24 true-online backup is used with user logging enabled:

```
Access is enabled.
   .
   .
Logging is enabled.
   .
   .
Database last stored using True-Online Backup and
   log file NLOG007 on THU, JAN 18, 1996,  6:06 PM.
```

  In this case, you may purge log files NLOG001 up to NLOG006. With the AUTO option of the GETLOG command, log files up to NLOG999 will be automatically created when needed. After that, the log file NLOG001 is automatically created and used as is done today. If the log file, next in sequence to be automatically created, already exists, DBOPEN will fail. As the log files are automatically created in a round-robin fashion as done today for the AUTO option, you will have to take extra measures to ensure that the log files, logically related to your database for recovery, are properly maintained. One way is to always start a new log cycle beginning with the first log file, NLOG001, of the log file set.

Error Message Enhancements

- **Broken Chain Error**

  The existing error message is:

```
BROKEN CHAIN - FORWARD AND BACKWARD POINTERS NOT CONSISTENT
```

  The new error message is:

```
    BROKEN FORWARD CHAIN:  SET # PATH # ENTRY # #
```

  or

```
    BROKEN BACKWARD CHAIN:  SET # PATH # ENTRY # #
```

  Due to current restrictions on DBERROR, the 32-bit entry number is split into two 16-bit numbers in the display. If the DBGET is first in the chain, the entry number is 0 0.

  The status register values are:

```
Status 16-bit Word 0   --> Condition Code
                   1   --> Set Number
                   2-3 --> Record #
                   4   --> 0 - Forward, 1 - Backward
                   5   --> Path #
                   6-7 --> Address of Base
                   8-9 --> Forward or Backward Record #
                           (Mode 5 - FWD, 6 - BWD)
```

- **Bad Record Pointer Error**

  This error is returned under the following circumstances:

  - DBGET mode 5 - when the user has just done a DBFIND and the forward chain pointer in the master record is invalid and outside of the detail data set capacity.
  - DBGET mode 6 - when the user has just done a DBFIND and the backward chain pointer in the master record is invalid and outside of the detail data set capacity.

  The existing error message is:

```
    INTERNAL TurboIMAGE/XL ERROR RETURNED ( -0307 )
```

  The new error message is:

```
    Bad record pointer for path # in set #
```

The status register values are:

```
Status 16-bit Word  0   --> Condition Code
                    1   --> Path #
                    2-3 --> Record #
                    4   --> 0
                    5   --> Acc.  Mode, PCODE
                    6-7 --> Address of Base
                    8   --> Mode of Intrinsic
                    9   --> Set #
```

- **No Chain Head Error**

  This error is returned under the following circumstances:

  - DBPUT into a detail with no corresponding manual master entry.
  - CIUPDATE into a chain with no corresponding manual master entry.

  The existing error message is:

  ```
  For DBPUT,
  No chain head (master entry) for path #

  For DBUPDATE,
  DBUPDATE:  No chain head (master entry) for path #
  ```

  The new message is:

  ```
  a) For DBPUT,
  No chain head (master entry) for path # in set #
  b) For DBUPDATE,
  DBUPDATE: No chain head (master entry) for path #
  in set #.
  ```

  The status register has one change in each case:

  For DBPUT: Previously unused, status word 1 = Set number

  For DBUPDATE: Previously unused, status word 3 = Set number

- **Lost Space in Dataset (TurboIMAGE/XL abort)**

  The existing error message is:

  ```
  LOST FREE SPACE IN DATA SET #.
  ```

  The error message is unchanged for master data sets:

  ```
  LOST FREE SPACE IN DATA SET #.
  ```

  But for detail data sets, the new error message is as follows:

  ```
  LOST FREE SPACE IN SET #, ENTRY # ACTIVE.
  ```

Other Enhancements

- **Enhanced Database Integrity**

  The DBPUT and DBUPDATE commands check the integrity of neighboring entries to detect possible data corruption before inserting into a chain.

- **DBUTIL**

  DBUTIL is now a native-mode program. In addition, a new command, `DETACH`, is added to detach the database from all DBEs the database is attached to. The new command is `DETACH` *dbname*.

- **DBSCHEMA**

  DBSCHEMA is now a native-mode program.

HP Business BASIC

by Sue Meloy
Support Technology Center

Introduction

HP Business BASIC/iX version A.00.15 is now available for MPE/iX Release 5.5. HP Business BASIC/V version A.02.14 is also available for MPE/iX Release 5.5 in compatibility mode. These versions incorporate several defect fixes and two enhancements.

The Business BASIC `HELP NEWS` command describes the contents of this release in more detail, including the list of defects fixed. Several aborts and instances of incorrect runtime behavior have been fixed with this release.

Enhancements

The Business BASIC `HELP ENHANCE` command describes these enhancements in more detail.

New ASSIGN Options

Two new options have been added to the `ASSIGN` statement to tell Business BASIC how to handle the carriage-control byte for MPE files created with the `CCTL` attribute. These new options have no effect if applied to a non-CCTL file.

The `CCTL` option to `ASSIGN` allows you to handle the carriage-control explicitly. On input, the carriage-control byte is returned as part of the input data. On output, the carriage-control byte is set from the first byte of the output data for each record.

The `NOCCTL` option to `ASSIGN` allows Business BASIC to handle the carriage-control independent of the user data. On input, the carriage-control byte is not part of the input data. On output, the carriage-control byte is set by Business BASIC independent of the output data. Examples:

```
10 ASSIGN #1 TO "cctlfile",NOCCTL
20 ASSIGN #2 TO "cctlfile",CCTL
```

ADDRESS Built-In Function

Business BASIC now provides a built-in function, `ADDRESS`, that returns the byte address of a Business BASIC variable.

This function can be used to construct records containing pointers, which may then be passed to routines written in other languages. For example:

```
10 Ptr = ADDRESS(Var)
```

Note

For Business BASIC/V, byte addresses may need to be converted to word addresses if the called routine expects word addresses.

C/iX Library Function getenv Change in Behavior

by Walter Murray
Support Technology Lab

Beginning with MPE/iX Release 5.5, the **getenv** function behavior has changed as a result of a defect fix in the C/iX Library.

Note

The **getenv** function change has the potential to affect existing programs that use the C/iX Library. It does not affect the POSIX/iX Library, so programs compiled and linked with the **c89** command of the MPE/iX Shell are not affected.

In the original implementation of the C/iX Library, the **getenv** function called **malloc** to allocate space for the retrieved value of the requested environment variable. The value of the pointer returned by **malloc** was then returned by **getenv** to its caller. The space allocated by **malloc** was never explicitly freed, so the heap could be exhausted after several million calls to **getenv** from the same process.

Starting with version A.05.12 of the C/iX Library, in MPE/iX 5.5, the **getenv** function uses a more conventional algorithm. It copies the value of the environment variable to a static buffer that is private to **getenv**, and then returns a pointer to that buffer. Each call uses the same buffer, so a value retrieved by one call to **getenv** is overwritten by a subsequent call.

The new implementation of the **getenv** function conforms with the ANSI standard for C, but it can cause a change in behavior for a program that takes advantage of the previous algorithm.

Following is an example of a program that is designed to display the current user and account names:

```
#include <stdio.h>
#include <stdlib.h>
main(void)
{
   char *user_ptr = getenv("HPUSER");
   char *acct_ptr = getenv("HPACCOUNT");
   printf("%s.%s\n", user_ptr, acct_ptr);
   return EXIT_SUCCESS;
}
```

This program is invalid, in that it takes advantage of the old implementation. It fails when using the new version of the library, because the second call to the **getenv** function overwrites the string returned by the first call.

The following version of the program corrects the problem.

```
#include <assert.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char *simulate_old_getenv(const char *name)
{
   /* Call getenv() with the name of the requested
      variable; then make a private copy of the
      string returned.  */
   char *p1, *p2;
   p1 = getenv(name);
   if (p1 == NULL)
      return NULL;
   p2 = malloc(strlen(p1) + 1);
   assert(p2 != NULL);
   strcpy(p2, p1);
   return p2;
}
main(void)
{
   char *user_ptr = simulate_old_getenv("HPUSER");
   char *acct_ptr = simulate_old_getenv("HPACCOUNT");
   printf("%s.%s\n", user_ptr, acct_ptr);
   return EXIT_SUCCESS;
}
```

Based on a survey of a number of users of C/iX, we feel that not many programs take advantage of the previous behavior of the **getenv** function. Therefore, this change should not be a significant obstacle to forward compatibility.

Pascal/iX Run-Time Library Heap Changes

by James Overman
Support Technology Center

Overview

This article describes changes in the Pascal/iX Runtime Library (HP31502) as of version A.05.01.07 in MPE/iX Release 5.5. The changes should have little impact on most programs except that they improve the internal accesses to the heap so that fewer processor cycles are required. The changes take effect immediately upon installation. No recompilation or linking is needed.

Features and Benefits

Following are the features and benefits of the changes:

- Heap blocks are allocated in larger chunks.
- Loss of small slivers of heap due to alignment options has been corrected.
- Execution performance of the Pascal heap routines has been optimized.

For any program that may be negatively impacted by the first change, there is a mechanism defined to revert to the old heap block allocation algorithm. For more details, read the remainder of this article.

Background

The MPE/iX Heap is used by many languages and utilities as well as the operating system itself. Recently, a few sites have reported problems with programs that have run out of heap space in certain conditions. Investigation into the heap and its allocation algorithms has determined that there were minor issues in the manner in which the heap was managed. Also, the mechanisms by which the heap was maintained were not well documented. This article is an attempt to clarify some of the heap mechanisms and to describe some changes that have been made to the heap routines to improve the general functionality of the Pascal/iX Heap.

Heap Structure

The general structure of the heap must be understood. For a given process, one fourth of the data address space is available for the heap space. Some of this quadrant is used for heap data structures and so not all the pages are available for the heap. The default heap size is configurable for the system and for each program. The standard heap size is 81,920,000 bytes or 40,000 pages of 2048 bytes of which about 39,964 pages are actually available. The minimum heap is 524,288 bytes and the maximum is 1,046,847,488 (around 511,160 usable pages).

Generally, the pages of the heap are not defined in transient space until they are actually used by the program. Do not just assign the maximum heap size to a system or program, as a rogue program that goes into a loop consuming heap space would run for a much longer time before the available heap was consumed (or all transient disk space was used).

The heap area is actually used to support three different heap data structures:

C language heap	Accessed via `malloc`, `realloc`, and `free`.
Pascal language heap	Accessed via `new`, `dispose`, `mark`, and `release`.
Other heap	Accessed via the intrinsics `P_GETHEAP` and `P_RTNHEAP` (and the older `GETHEAP` and `RTNHEAP`).

These three heaps are maintained independently of one another but must ultimately share the one heap area of the process.

Programs may legitimately run out of heap. If a program consumes all of the available heap, it is normally reasonable to just increase the NMHEAP size on the `RUN` command or to do an `ALTPROG` in the Link Editor on the program file. Just because the program runs out of heap in a Pascal heap call, does not mean that the majority of heap was not consumed by the C heap, and vice-versa.

If a program has a problem with the Pascal heap, you should verify that heap compaction and disposal have been turned on (the defaults are off). `$HEAP_COMPACT ON$` and `$HEAP_DISPOSE ON$` should be in the compiled code.

Performance of programs using the heap is sometimes the fastest when the heap Compaction and Disposal are OFF, assuming that the program does not consume immense amounts of heap and cause the system to begin to swap memory pages. The performance effect of turning Compaction and Disposal ON may vary from insignificant to very significant, positively or negatively. Only actual user testing and analysis will tell for a given program and system.

Heap Request Processing

In the normal processing of heap calls, a request for a specific amount of memory activates a search of the free space list for the given heap. If found, the appropriate amount of memory is removed from the free space and a pointer to the assigned space is returned to the user. If adequate free space is not found, a request for a portion of the heap area to be allocated to the specific heap (C, Pascal, Other) is made. The space provided is then added to the free space list and the search for the requested amount is repeated. If the required space is not available, the request is rejected and typically the program aborts with an error of `Out of Heap` or `System resource exhausted`.

Block Size From the Heap

The sizes of the blocks allocated from the heap area depend on the specific heap and other factors. For the Pascal heap, the size of the request is rounded up to a multiple of the page size. Then, depending upon the number of previous requests for heap area, a minimum block size is calculated.

If the minimum block size is larger than the rounded request, the minimum block size is used for the request for heap area. The minimum block size was set to one page for the first request, two pages for the second through the seventh request, five pages for 8th through 15th, 10 pages for the next 10 requests, 20 for the next 12 requests, and 40 pages for all subsequent requests. Thus, the Pascal heap was divided into several modestly sized blocks and any larger blocks as needed by large user requests.

Multiple Blocks

The division of the heap into multiple blocks has an advantage and disadvantage.

- The advantage is that the division of the heap area into multiple blocks can shorten the scan for free space significantly (depending upon your application).

  Each Pascal block has its own free space list which is maintained in decreasing size order. Thus a request for a certain size causes a scan of the free space list of the newest block, if that block does not have enough free space (which is determined simply by checking the first free space, as it is the largest), then the next block is checked, and so on. But, if a block does have a large enough free space, the remaining free space list is scanned until the smallest free space area is found that will satisfy the request in that block.

- The disadvantage is that the total free space is divided among the blocks and cannot be compacted (as only the space within a block is compactible).

  Because the space cannot be compacted, a heap fragmentation caused by the block allocation algorithm may cause heap requests to be rejected even when total free space might exceed the requested size.

  This fragmentation can be somewhat reduced by using the Pascal Mark and Release mechanism. The Mark request effectively starts a new Pascal heap that is in addition to any previous Pascal heap. Space requests are filled from this new heap until the next Mark or until a Release call frees all of the space that has been allocated since the related Mark. The Released blocks are actually returned to the heap area and become available for future requests. Any files opened that reference the Marked spaces are closed when the spaces are Released.

  The Other heap is allocated in blocks like the Pascal heap and those blocks may be interspersed with the Pascal heap blocks. These Other heap blocks are never released and so contribute to the heap fragmentation.

The C heap allocation algorithm is such that the C heap grows from the opposite end of the heap area that the Pascal and Other heaps do. As the opposite heaps grow towards each other a "Pascal high water mark" is maintained to assure that the heaps do not collide.

Pascal/iX Library Algorithm Enhanced

Enhancements to the Pascal and Other heap routines in the Pascal/iX Library version A.05.01 and later is an attempt to improve the fragmentation problem. Briefly, the enhancements are:

- Minimum block algorithm has been enhanced to reduce the number of blocks.
- Dumping Pascal and Other heap information is available.
- CPU performance has been improved.
- Heap allocation and compaction routines defect was corrected.
- `P_GETHEAP` routine has been enhanced to consolidate the newest allocated block with the prior free block.

These Pascal/iX heap changes will not be noticeable to most users. Actual heap usage may be slightly more in a few cases, but should be less in other cases due to the reduced fragmentation. Those programs that are negatively impacted may use the new CI variable to revert to the old allocation algorithm or to specify optimal block sizes for their needs.

The following sections discuss these changes.

Minimum Block Algorithm Enhancement

The minimum block algorithm has been enhanced to be one page for the first request, 2 pages for the second, then 4, 8, 16, 32, 64, 128, and 256 for all subsequent requests, until there is not enough space for the large block. Then a block size just big enough to fulfill the request is allocated.

The intent is to reduce the number of blocks for those applications that use a lot of smaller chunks of heap. This reduction in blocks should help, but may hinder some programs (although testing to date shows improvements). For those that it negatively impacts, an alternative has been developed.

A Command Interpreter Variable named `HP_PASCAL_MINIMUM_HEAP_BLOCK` may be set to override the allocation algorithm. A positive value becomes the minimum block size (in pages) that the allocation request uses. Thus, programs with very large heap needs might use 1024 or 4096 pages for improved performance. If a value of 0 is specified, then the old allocation algorithm is used (1,2 for 6 times, etc. until the maximum of 40 pages). Negative values invoke the new default allocation algorithm.

Dumping Heap Information Available

For those who want to know more about their heap usage, the routine named P_DUMP_HEAP (with no parameters) may be called to produce a dump of the Pascal and Other heap structures. The P_DUMP_HEAP routine is available on earlier versions of the library; the output has been expanded for version A.05.01.

Also, in A.05.01, the Pascal normal termination routine (P_TERMINATE) has been enhanced to check for a CI Variable named HP_PASCAL_HEAP_DUMP. If set to 1, P_DUMP_HEAP is called to produce a heap printout. If set to greater than 1, the P_DUMP_HEAP plus information about the entire heap is output. The information provided by this feature is intended to facilitate the isolation of heap problems with the Response Center, Expert Center, and the Lab, so the format and information may change with time.

CPU Performance Improved

The CPU performance has been improved, as code optimization has been performed based on data collected and analyzed by SPT/iX for some test programs.

Defect Corrected

A defect in the heap allocation and compaction routines has been corrected. The defect was causing small slivers of the heap to be lost when alignment values greater than four bytes were used.

P_GETHEAP Routine Enhancement

The P_GETHEAP routine was enhanced to consolidate the newest allocated block with the prior block when the prior block is entirely free. This should help those programs that acquire space via P_GETHEAP, return it, and then request a larger space than before. This special case should not impact the vast majority of programs as the enhancement is very case specific.

Inform/V Contains Several Defect Fixes

by Bill Toms
Support Technology Center

Introduction

This release of Inform/V version A.10.00 (HP32246) in MPE/iX-Express 3 based on General Release 5.0, contains 12 defect fixes. These fixes are on the PowerPatch tape (C.50.03). The fixes may affect your current set of reports.

The following paragraphs describe the corrections made to Inform/V for the highest priority problems reported by customers. Please refer to the Software Release Bulletin for more details on these fixes and all the other fixes.

Note To take advantage of some of the changes, you may need to re-create or modify affected reports.

Different Driving File May Be Selected

Inform/V now selects the driving file as shown in the Inform/V Reference Manual. Be aware that your reports may change if your Inform groups contain elements with the same link values and/or no file owners. The previous versions of Inform/V (A.08.05) selected the first possible file as its driving file. It has been changed to select the best file for its driving file. You may need to make adjustments in your Inform groups via DICTDBM if reports are not displaying the expected information.

See the Inform/V Reference Manual for information on creating Inform groups and selecting a driving file. Similar information is also included in the Dictionary Reference Manual.

SET OPTIONS Menu Changed

The SET OPTIONS menu in Inform/V has changed. There is a new prompt for the display length of a calculated field. This allows you to avoid DECIMAL OVERFLOW errors on a user-defined item by expanding the item's definition to fit the data.

Note If you have batch jobs which access the SET OPTIONS menu, you will need to add another response.

Signed and Unsigned P and Z Type Key Items Considered Equal

Inform/V no longer differentiates between a signed and an unsigned positive number for key items of type P (packed) or Z (zoned). For example, selecting such an item with a value equal to 123 in the selection criteria will now find all those items equal to either 123 or +123. This can be very important since P and Z type item values entered using QUERY may include a plus sign.

Selection Processing Improvements

Some miscellaneous problems with selection processing have also been fixed. Eight and twelve byte integer fields now match properly even if they contain negative values. The wildcard characters "^^" now work correctly on two character fields and also when using languages other than NATIVE-3000.

Transact Contains Defect Fixes

by Bill Toms
Support Technology Center

Introduction

The release of Transact/iX version A.05.02 (HP30138) in MPE/iX-Express 3 based on General Release 5.0, contained 27 defect fixes. The compatibility mode version of Transact/V version A.10.02 (HP32247), contained an additional 28 defect fixes. These fixes are on the PowerPatch tape (C.50.03).

This article highlights some of the corrections made to each product. Please refer to the Software Release Bulletin for more details on these fixes and all the other fixes.

Note

For the MPE/iX 5.5 Release, three additional defects have been fixed in Transact/iX, version A.05.04 (HP30138).

TRANSACT/iX

The following paragraphs briefly describe the corrections made to Transact/iX for some of the highest priority problems reported by customers.

Compiler Needs Less Heap Space

The Transact/iX compiler now requires less NMHeap. This prevents compiles of most very large programs from aborting with an `out of memory` message. This became a problem for some programs starting with the A.05.00 release of Transact/iX.

ACI Closes Files

When a program written in another language such as COBOL calls a Transact/iX program using the Architected Call Interface (ACI), TRANIN and various TRANDEBUG files no longer remain open after exiting the Transact program.

NOMATCH Option in Serial Access

The match register is no longer checked when the `NOMATCH` option is specified in `FIND(SERIAL)`, `GET(SERIAL)` and `OUTPUT(SERIAL)` statements applied to a TurboIMAGE database.

PERFORM= Option of a DELETE(CURRENT)

The `PERFORM=` option of a `DELETE(CURRENT)` statement now executes the code following the specified label as expected. Previously, the jump to the label was not taken.

DISPLAY Verb

The NEED= option of the FORMAT statement is now being checked when DISPLAY is printing the last line of data on a page.

The trailing "–" in an *edit-string* now works consistently with the trailing "CR" for items of type "9".

The output should now be consistent between Transact/V and Transact/iX when specifying the LINE= option on a DISPLAY to a file.

KSAM File Usage

Several defects have been fixed related to KSAM files and dealing with incorrect or incomplete retrievals, incorrect matching, partial keys, and problems where the items are not specified in the same order as the physical record.

PASSWORD is Padded

The PASSWORD option of the LIST statement now pads with blanks, just as it once did.

TRANSACT/V

The following paragraphs are intended to give Transact programmers some insight into the areas of the product most affected.

DISPLAY Verb

The NEED= option of the FORMAT statement is now being checked when DISPLAY is printing the last line of data on a page.

The trailing "–" in an *edit-string* now works consistent with the trailing "CR" for items of type "9".

Multiple CCTL options in a FORMAT statement should now be applied to the correct item even if there are fewer CCTLs than items.

Using the ROW= option when displaying to a file now skips to the correct row in the file.

Use of DISPLAY and DISPLAY(FILE=*filename*) now keeps better track of line counts, page breaks and the need to issue a CONTINUE Y/N? prompt.

The page numbering when displaying to a file should now be correct even when that file has a blocking factor specified in the SYSTEM statement.

A CLOSE of a file should no longer result in an FSERR 0 if a DISPLAY has been done to that file.

LET Verb

Several error situations with the LET statement have been fixed. Also a LET of a zero from a numeric type to an X or U type now results in a single ascii zero instead of all blanks.

KSAM File Usage

Several defects have been fixed related to KSAM files and dealing with incorrect or incomplete retrievals, incorrect matching, partial keys, and problems where the items are not specified in the same order as the physical record.

PASSWORD in a Called Subprogram

Two defects have been corrected concerning the result of the following statement in a called subprogram where the password was a parameter in the CALL statement:

```
LIST item,PASSWORD;
```

Transact/V and Transact/iX are now more consistent with each other and the manual.

Data/Application Integration

Announcing DCE/3000

by Joe-Sue Wang
General Systems Division

Product Description

The Distributed Computer Environment (DCE) technology was originally designed by Open Software Foundation (OSF). The services that are provided in DCE are acknowledged by the industry as the most complete and comprehensive framework for data and applications to be distributed transparently across the networks and systems.

DCE/3000 has been an orderable product since May 1, 1995. The new version of DCE/3000 (version A.01.12), is available beginning with MPE/iX-Express 3 based on General Release 5.0. DCE/3000 version A.01.12 is still based on the OSF DCE 1.0.2 implementation. However, the DCE library is provided as both an archive (libdce.a) and an executable library (DCEXL.HPDCE.SYS).

The addition of DCE to the HP 3000 platform enables the DCE-based applications to run on the MPE/iX operating environment.

Supported Components

DCE/3000 provides all the core services that are required for a DCE cell to function correctly. A cell is the basic unit of operation and administration in DCE. The core services include:

- Remote Procedure Calls (RPC). They support the development of distributed applications by making requests to remotely networked machines as if they were local. RPCs also implement network protocols used by clients and servers to communicate with each other.
- Cell Directory Service (CDS). It manages a database for the resources in a group of machines called a DCE cell. The database consists of the names of resources and associated attributes.
- Distributed Time Service (DTS). It provides synchronized time for the computers in the distributed computing environment.

- DCE Security. It provides secure communications through use of services such as authentication which guarantees the identity of users, and authorization which keeps track of what privileges the user may be granted.

Order Information

DCE/3000 is an independent product that is not distributed on the SUBSYS tapes of system releases. However, it requires your system to be on the MPE FOS version C.50.02 or later. You can order the following DCE/3000 product family through your sales representatives:

Product Order Number	**Description**
B3821AA	contains DCE/3000 Core Services media and manuals; it is available to HP customers in the United States only.
B3822AA	contains DCE/3000 Core Services media and manuals; it strips the DES encryption algorithm and is available to all HP customers.
B3823AA	DCE/3000 Executive Client license
B3824AA	DCE/3000 Cell Directory Service (CDS) Server license
B3825AA	DCE/3000 Security Server license
B3827AA	DCE/3000 Manuals

8

Networking/Client-Server

Enhanced Console Switching

by Barbara Dubbert
Commercial Systems Division

Overview

The console command has been enhanced to allow console switching over the LAN. The system console can now be switched to virtual terminal (VT) sessions and sessions initiated from PCs and workstations. However, this enhancement does not support switching to PAD terminals.

Use of this new capability can improve system administration. Multiple systems in a network can switch their system consoles to the same terminal, PC, or workstation for a single point of control.

Return to LDEV 20

For session disconnects, the console automatically switches back to LDEV 20, the physical console. Switching to LDEV 20 occurs for both normal and unexpected logoffs. An unexpected logoff can be caused by an `ABORTJOB` against the system console session or by disconnects due to software errors.

Error Message Changes

When attempting to switch the console to an unsupported terminal type device, such as a PAD terminal, an error is displayed on the session issuing the console command. The console remains where it is currently assigned.

For the MPE/iX 5.5 Release, the error text has been changed to more accurately reflect the cause of the error:

```
:console 92
 ----------^

This device cannot be used as a console. (CIERR 3131)
```

NetWare Print Enhancements

by Rosemarie Chiovari
Commercial Systems Division

NetWare v3.11 for the HP 3000 has been enhanced to allow printing of MPE spoolfiles and other MPE files to LAN-connected printers via the SPX/IPX JetDirect card or printers, slaved off a NetWare client PC running RPRINTER.

The NetWare print daemon (nwpd) allows spoolfiles to be automatically routed to NetWare queues for printing on LAN-connected printers. The spoolfiles must be text files or files that have been formatted by an application for a printer.

MPE files may be directed to a LAN-connected printer via the `nwprint` command. Criteria for printing of MPE spoolfiles and `nwprint` parameters are specified in the required nwpd control file /SYS/NETWARE/.NWPDRC.

For details, please refer to the README.NETWARE.SYS file.

Network Printer Support Now Available

by Steve Bitondo and Shelley Nelson
Commercial Systems Division

Prior to MPE/iX Release 5.5, the Native Mode Spooler (NMS) allowed many programs to share a single printer connected directly to the HP 3000. With Release 5.5, the spooler now supports *network printers*, that is, any Printer Command Language (PCL)-based printers attached to the HP 3000 via a TCP/IP network connection and a JetDirect interface.

The system manager is responsible for preparing network printers for use with the HP 3000, which includes three tasks:

- Connecting and preparing the printer(s) according to the instructions furnished with the hardware and with the networking system.
- Using SYSGEN to add the network printer(s) to the MPE/iX I/O device configuration.
- Creating the NPCONFIG.PUB.SYS configuration file, which defines a variety of operating parameters used by the network printer(s). NPCONFIG must include an *LDEV-specific entry* for each network printer that you want placed in service, and it may include a *global entry* that defines parameters common to all network printers.

The table below lists all HP devices available for use in an HP 3000 network printing environment. Some of the devices are listed by family, such as "PaintJet". Specific exceptions, such as LaserJet 4L, are listed separately; the family designation then applies to the rest of the family. The table also indicates whether or not Page Level Recovery and Page Count Logging are supported for each device.

Supported Networked Devices

Device/ Family	PLR	Page count
Color LaserJet	Yes	Actual
LaserJet 4 family (except 4L)	Yes	Actual
LaserJet 4L	No	Estimate
LaserJet III family	No	Estimate
LaserJet II family	No	Estimate
HP5000/C30	No	Estimate
HP5000/C40, without PJL support	No	Estimate
HP5000/C40, with PJL support	Yes	Actual
PaintJet, DeskJet, QuietJet, ThinkJet family	No	Estimate

The *Native Mode Spooler Reference Manual* (32650-90166) has been updated for Release 5.5. It includes a new chapter, "Configuring and Operating Network Printers," that teaches members of the system administration staff how to configure and operate a network printer, and teaches other users how to work with network printers.

For more detailed information about network printing in this *Communicator*, read the "Network Printing Technical Overview" article in Chapter 10, "Technical Articles."

Introducing Internet Services on the HP 3000

by Steve Bitondo and Shelley Nelson
Commercial Systems Division

The Internet Services

Internet Services will be available to HP 3000 customers for the first time with version C.55.00 of MPE/iX. These services consist of a set of programs that help the HP 3000 exchange information with other nodes on an internet.

The Internet Services offered on the HP 3000 are a subset of the Internet Services available on the HP 9000, which was previously called the ARPA Services. The services are briefly described in the table below.

Summary of HP 3000 Internet Services

Service	Description
inetd	The Internet daemon **inetd** is the master server for the group of Internet Services rather than an individual network service. You must install and configure **inetd** on your system to use the other services, **telnet**, **bootpd** and **tftpd**.
Telnet	The Telnet server uses the standard virtual terminal protocol to allow users on a remote node that supports Internet Services to log on and run most applications on the host HP 3000.
bootpd	The Bootstrap Protocol daemon, or **bootpd**, is used to boot, or start, devices such as routers, printers, X-terminals and diskless workstations. Client systems use **bootpd** to find their own IP address and the name of the boot file to load into memory and execute.
tftpd	The Trivial File Transfer Protocol daemon **tftpd** is used to transfer the boot files needed to start network devices. In this implementation of Internet Services, **tftpd** enables an HP 3000 to boot network printers.

All of the Internet Services program and configuration files come with version C.55.00 of the MPE/iX Fundamental Operating Software (FOS). Part of this software, the Telnet client (TELNET.ARPA.SYS), was first made available to customers on the C.50.00 version of MPE/iX. As part of MPE/iX FOS, Internet Services can run on any Precision Architecture-RISC model of the HP 3000. They are not available on earlier "classic" HP 3000 computers running MPE V.

Configuring the Internet Services

The system manager is responsible for the installation and configuration of the Internet Services. This procedure requires:

- The installation of one or more network interface link cards that support TCP/IP communications protocol. At least one such card is delivered with each PA-RISC HP 3000 system.
- The installation of the Net Transport communications software which uses the TCP/IP protocol. Internet Services runs on top of the Net Transport software and therefore runs over any type of link supported by Net Transport.
- Creating new configuration files (from the sample files installed in the NET group of the SYS account) and editing the files to configure the Internet Services. Or, editing the configuration files you already have to add the new Internet Services to your system.

To use the Internet Services, you must, at a minimum, configure the configuration file for `inetd`. Then for each of the individual Internet Services you wish to add, you must also edit any other configuration files required by the specific service. This procedure is explained in the new manual *Configuring and Managing Internet Services* (32650-90835) which is available with version C.55.00 release of MPE/iX.

Introducing the Telnet/iX Server

by Debbie Cooper and Cas Caswell
Commercial Systems Division

Overview

Telnet/iX has been enhanced on MPE/iX version C.55.00 to provide Telnet server functionality. The availability of both the Telnet/iX Server and Telnet/iX Client (previously released on MPE/iX version C.50.00) strengthens the HP 3000 as an open system by providing access between the HP 3000 and computers that support Telnet, such as UNIX-based systems and PCs.

The Telnet protocol provides virtual terminal capability across a network supporting TCP/IP (Transmission Control Protocol and Internet Protocol). Telnet and FTP (File Transfer Protocol) were developed by the University of California, Berkeley for the Advanced Research Projects Agency (ARPA).

On previous versions of MPE/iX, FTP and the Telnet/iX Client were identified as ARPA Services on the HP 3000. Now with MPE/iX version C.55.00, Telnet/iX and FTP are considered part of a larger set of services known as the MPE/iX Internet Services. For more information, read the article, "Introducing Internet Services on the HP 3000," in this chapter.

Using the Telnet/iX Server

The Telnet/iX Server allows an HP 3000 "host" system to accept logons from remote "client" systems. The client can be an HP 3000, HP 9000, PC or any computer that is connected to the network and is using the Telnet protocol via TCP/IP.

The Telnet/iX Server functionality is available in two parts:

- With MPE/iX Release 5.5, the Telnet/iX Server allows a user to logon to the HP 3000 and use all MPE/iX CI commands as well as issue some Telnet client commands to the Server.
- Full functionality, as described in the 5.5 version of the manuals listed in the "Documentation" section later on, will be available in Fall of 1996.

Please see the article, "Telnet/iX Server Functionality Details," for detailed information on the functionality available with MPE/iX Release 5.5.

Important Details Please Read

On UNIX systems, there is one process called **telnetd** for each inbound Telnet connection. However, when using the Telnet/iX Server, the remote user's session is managed with the standard session processes JSMAIN and CI. This implementation eliminates the overhead incurred with an additional process for each Telnet connection. JSMAIN, CI, and any user applications run from the CI.

communicate to the Telnet/iX server driver (part of NL.PUB.SYS) through the MPE file system.

Security Checking

The security checking method available on MPE/iX is provided by the Internet daemon's internal security. To implement security checking, you must edit the `inetd` security file so that specific nodes are allowed or denied Telnet access to your system. For information, read "The inetd security file" in Chapter 2 of the *Configuring and Managing MPE/iX Internet Services* manual (36957-90154).

DTC Telnet Access

Prior to MPE/iX version C.55.00, Telnet server functionality on an HP 3000 was only available with DTC Telnet Access. DTC Telnet Access processes incoming Telnet connections via a DTC configured with a Terminal Access Card (TAC) using OVDTCMGR, the PC-based manager. Unlike DTC Telnet Access, the Telnet/iX Server requires no additional hardware, resulting in lower cost and lower configuration and management maintenance. Although the Telnet/iX Server provides similar functionality, DTC Telnet Access will continue to be offered to customers with high Telnet traffic to minimize the load to the host HP 3000 CPU.

Telnet/iX Server Requirements

The Telnet/iX Server requires version C.55.00 of MPE/iX Fundamental Operating System and a TCP/IP supported network link. Links supported on MPE/iX include:

- Ethernet/802.3
- Token Ring/802.5
- FDDI
- X.25

To install The Telnet/iX server, the system manager must update the following two files.

- SERVICES.NET.SYS

  The Internet Services file that associates an official service name and alias with the port number and protocol the service uses.

- INETDCNF.NET.SYS

  The configuration file that must be updated to instruct `inetd` to listen for Telnet connection requests. Inetd, the master server for the Internet Services, continually monitors incoming requests for services configured on the host HP 3000.

Documentation

The following customer documentation provides information on the Telnet/iX server:

- *HP Telnet/iX User's Guide* (36957-90154)

  This manual, formerly called *HP Telnet/iX Client User's Guide* (36957-90152), is the user manual for both Telnet/iX Client and Server. It is distributed with the System Management Core A (FOS) Manual Set (36367A).

- *Configuring and Managing MPE/iX Internet Services* (32650-90835)

  New with version C.55.00 of MPE/iX, this manual describes how to configure and operate the Internet Services on the HP 3000. It is distributed with the System Management Core A (FOS) Manual Set (36367A).

- *Asynchronous Serial Communications Programmer's Reference Guide* (32022-61001)

  This manual provides information regarding restrictions and special considerations with use of the MPE/iX file system intrinsics and the Telnet/iX Server. It is distributed with the MPE/iX Programming Core Plus Manual Set (36370A).

DTS/TIO Dynamic Configuration and Host-Based Switching

by John Spitzer
Commercial Systems Division

Introduction

The Datacommunications and Terminal Subsystem (DTS) consists of the software that resides on the HP 3000 and the Datacommunications and Terminal Controller (DTC) family of terminal servers that are used to provide LAN-based, asynchronous connections to HP 3000 systems.

The DTC family of products also provides X.25 communication and Telnet access to HP 3000 systems, and TCP/IP communication to HP 9000 and non-HP systems. The DTS allows terminals, PCs in terminal emulation mode, serial printers, and other asynchronous devices to communicate with LAN-based hosts in HP-only and multivender networking environments.

Features and Benefits

Features

The MPE/iX Release 5.5 provides the following new product features:

- Dynamic configuration of DTS/TIO devices without rebooting the HP 3000 host.
- Automatic configuration of new DTC servers that eliminates almost all configuration steps.
- Shutdown and restart of the DTS without rebooting the HP 3000 host.
- Terminal switching on Host-based DTC ports.
- Back-to-back Configuration on Host-based DTC ports.
- Domain Name Service and IP routing configurable on Host-Based DTCs.
- Configuration of Forced Data Forwarding on Host-based PAD terminal ports.
- Support of 1024 profiles.
- Capability to specify the starting LDEV for TIO and PAD non-nailed LDEV pools.

Benefits

With these DTS/TIO enhancements to MPE/iX 5.5, system administrators and end-users will find:

- HP 3000 provides fulltime availability for end-users even when terminal I/O configuration changes are needed.

- The system no longer needs to be shutdown to make DTS/TIO changes.
- Most DTS/TIO configuration changes do not affect other TIO users.
- Improved ease of use when configuring new DTCs.
- Improved connectivity when using host-based DTC configuration.

These enhancements significantly improve the system availability and reduce the need for planned down time.

Documentation

For more details, refer to:

- The technical article, "DTS/TIO Dynamic Configuration and Host-Based Switching Technical Overview," in Chapter 10, "Technical Articles."
- The manual, *Configuring System for Terminals, Printers, and Other Serial Devices* (32022-61000).

9

Peripherals

Online System Device Configuration

by Rick Ehrhart and Rakesh Patel
Commercial Systems Division

Online Device Configuration is a new MPE/iX facility that allows system managers and operators to configure and deconfigure new devices such as tape drives, disks and system printers while the system is online. This new facility eliminates rebooting of the system in conjunction with device configuration changes and therefore increases the system availability.

The system manager or operator can configure devices online in one of two ways:

- Using the new IOCONFIG utility
- Using the I/O configurator in the SYSGEN utility, and then issuing the new `DOIONOW` command

Using IOCONFIG

IOCONFIG is a new utility created specifically for configuring disks, tape drives and system printers online. It has a command interface identical to that of the SYSGEN I/O configurator. IOCONFIG can completely replace the use of SYSGEN for device configuration because IOCONFIG automatically updates SYSGEN's base configuration.

Any changes that the system manager or operator makes with IOCONFIG take effect immediately. As a result, you should continue to use SYSGEN to configure devices when you want the changes to take effect at the next reboot. In other words, continue to look upon SYSGEN as the static configuration tool, while relying on IOCONFIG to do the online configuration of devices.

You may use IOCONFIG in one of two ways: interactively or non-interactively, (which is sometimes called "command mode").

To use IOCONFIG interactively, you issue the `IOCONFIG` command, without command parameters, at the CI prompt. This starts the IOCONFIG utility, at which point you may enter any of the commands shown in the following table at the special prompt. When you are through using IOCONFIG, you must explicitly exit the utility.

IOCONFIG Commands

Command	Abbreviations	Description
`ADDDEVICE`	`adev, ad`	Adds a device from the physical configuration to the active configuration.
`LISTDEVICE`	`ldev, ld`	Lists the active device configuration.
`DELETEDEVICE`	`ddev, dd`	Deletes a device from the active configuration.
`ADDCLASS`	`aclass, ac`	Adds a device class to the active configuration.
`MODIFYCLASS`	`mclass, mc`	Modifies a device class in the active configuration.
`LISTCLASS`	`lclass, lc`	Lists the active device class configuration.
`DELETECLASS`	`dclass, dc`	Deletes a device class from the active configuration.
`ADDPATH`	`apath, ap`	Adds an intermediate path to the active configuration.
`LISTPATH`	`lpath, lp`	Lists I/O paths for the active configuration.
`DELETEPATH`	`dpath, dp`	Deletes an I/O path from the active configuration.
`REDO`	`redo`	Re-executes the command previously executed.
`HELP`	`he, h`	Displays information about all commands or about the specific command entered following the `HELP`.
`EXIT`	`ex`	Exits IOCONFIG.

To use IOCONFIG in command mode, you enter `IOCONFIG` followed by a single command passed as an INFO string at the CI prompt. For example, you might want to list the devices or device classes in the current configuration without executing any other command. To do so, you would enter:

```
:IOCONFIG "lc"
```

Device and device class configuration with the IOCONFIG utility is just like SYSGEN's. You can list device and device class information, or add and delete a device or a device class. You can also modify an existing device class to add or delete devices in the class, or to rename the class. All commands take effect immediately. As a side-effect, those commands that update the configuration also update the current SYSGEN boot configuration.

Adding a Device

To configure a tape drive, disk or a system printer into the system, the system manager or operator uses the ADDDEVICE or ADEV command. The syntax of the command is identical to that of SYSGEN's ADEV command in IO level, except that the ID parameter is optional. The command syntax is as follows:

```
ADDDEVICE; { [LDEV =]#/#,#, ... } { [PATH =]devicepath}

[ [ID =]productid]   [ [RSIZE =]recordsize]
[ [OUTDEV =]outputdevice]

[           ( JOB          ) ]
[           ( DATA         ) ]
[           ( INTERACTIVE  ) ]
[           ( DUPLICATIVE  ) ]
[ [MODE =]  { INPUT        } ]  [ [CLASS =]classname]
[           ( OUTPUT       ) ]
[           ( AUTOREPLY    ) ]
[           ( NLIO         ) ]
[           ( NONE         ) ]
[           ( IN      ) ]
[           ( OUT     ) ]
[ [CMODE =] { CIO     } ]
[           ( NCIO    ) ]
[           ( RANDOM  ) ]
[           ( DEFAULT ) ]

[ [PMGR =]physicalmanagername]

[ [LMGR =]logicalmanagername]

[ [PMGRPRI =]physicalmanager priority]

[ [MPETYPE =]compmodetype]

[ [MPESUBTYPE =]compmodesubtype]

[ [DEVNAME =]devicename]
```

To configure a SCSI device, it must be physically connected to the specified path, it must be operational, and there must be a device-defaults-data entry for the product ID (which is obtained from the device during configuration). If you omit the ID parameter, the IOCONFIG utility interrogates the hardware to identify the device. It then validates the ID that it retrieves by checking it against the list of supported devices.

When adding a printer to the configuration that has the class SPOOL, the spooler process for that device will automatically be started and a message indicating this will appear on the console.

Listing One or More Devices

The system manager or operator can list all devices in the system or one or more specified devices along with their configuration information using the LISTDEVICE or LDEV command. When you enter either command without parameters, information about all devices currently configured in the system is listed. By providing one or more parameters as the selection criteria, you can see information about particular devices.

The command syntax is as follows:

```
LISTDEVICE;  [[LDEV]= #/#,#, ... ]
             [[ID]= product number]
             [[TYPE]= device type]
             [[CLASS]= classname, ... ]
             [[DEST]= OFFLINE]
```

Deleting a Device

The system manager or operator may delete a device from the configuration using the DELETEDEVICE or DDEV command. A device is considered to be completely deconfigured when all its system-related resources are recovered. DDEV, at this time, can only deconfigure two kinds of devices completely: network printers and disks that are attached to single-ended SCSI adapters.

The syntax of the DDEV command is identical to that of SYSGEN's DDEV command in IO level. The command syntax is as follows:

```
DELETEDEVICE;  {[LDEV =]#/#,#, ... }
               [[ID =]device id]
               [[TYPE =]device type]
               [[CLASS =]classname]
```

Adding a Device Class

To create a new device class in the system, the system manager or operator uses the ADDCLASS or ACLASS command. The syntax of the ACLASS command is identical to that of ACLASS in SYSGEN. All devices that you designate as members of the class (using the LDEV parameter) must be configured before issuing this command.

The command syntax is as follows:

```
ADDCLASS  { [CLASS =]classname }  { [LDEV =]#/#,#, ... }

                     ⎡          ⎧ IN      ⎫ ⎤
                     ⎢          ⎪ OUT     ⎪ ⎥
                     ⎢ [MODE =] ⎨ CIO     ⎬ ⎥
                     ⎢          ⎪ NCIO    ⎪ ⎥
                     ⎢          ⎪ RANDOM  ⎪ ⎥
                     ⎣          ⎩ DEFAULT ⎭ ⎦
```

Modifying a Device Class

The system manager or operator can modify a device class by adding or deleting one or more devices in it or by renaming it using the MODIFYCLASS or MCLASS command. The syntax of this command is identical to that of MCLASS in SYSGEN. The command syntax is as follows:

```
MODIFYCLASS  { [CLASS]= classname }

             [ [NEWCLASS]= classname ]

             [ [ALDEV]= logical device #, ... ]

             [ [DLDEV]= logical device #, ... ]

             ⎡          ⎧ IN      ⎫ ⎤
             ⎢          ⎪ OUT     ⎪ ⎥
             ⎢ [MODE =] ⎨ CIO     ⎬ ⎥
             ⎢          ⎪ NCIO    ⎪ ⎥
             ⎢          ⎪ RANDOM  ⎪ ⎥
             ⎣          ⎩ DEFAULT ⎭ ⎦
```

Currently, the system allows any device in the system to belong to only one associated class at any time. For example, if a device X belongs to a class Y and Y is associated to some user, then the MODIFYCLASS command prevents you from adding X to another class Z which is also associated to any user at the time.

Listing One or More Device Classes

The system manager or operator can list all device classes in the system, or one or more specified device classes, with their configuration information using the LISTCLASS or LCLASS command. When no parameter is used with this command, all device classes currently configured in the system are listed. By providing the names of those device classes with the CLASS keyword, you can view information about particular device classes.

The command syntax is as follows:

```
LISTCLASS    [[CLASS]= classname, ... ]
;               [[DEST]= OFFLINE]
```

Deleting a Device Class

The system manager or operator can use the DELETECLASS or DCLASS command to delete a device class that is not currently associated to a user. To disassociate the device class from the user, the user must execute the DISASSOCIATE command at the Command Interpreter prompt.

The command syntax is as follows:

```
DELETECLASS  {[CLASS =]classname}
```

Configuring a Device Adapter

It is often necessary to configure intermediate paths before a device can be configured into the system. For example, you must configure a SCSI device adapter before the first device on that SCSI bus can be configured. Intermediate path configuration is done just like in SYSGEN. At this time, the command to delete a path is only supported for the single-ended SCSI adapter. All other cases will fail with an error.

The command syntax is as follows:

```
ADDPATH;  {[PATH =]devicepath}  {[ID =]productid}
           [[PMGR =]physicalmgrname]
           [[PMGRPRI =]physicalmgrpri]
           [[LMGR =]logicalmgrname ]
           [[MAXIOS =]maxconcurrentchannelIOs]
```

Listing an I/O Path

An I/O path is the system address assigned to the device interface hardware and the physical path used to reach an I/O device. The system manager or operator can use the LISTPATH command to display information about adapters and I/O devices on a specified path in the active configuration. The command lists the configured I/O paths according to their paths or to their associated I/O manager. The syntax of the LISTPATH command appears below:

```
         ┌PATH= [path]                 ┐
LISTPATH │LEVEL= [#]                   │
         │MANAGER= [manager name]      │
         └DEST= [OFFLINE]              ┘
```

The LEVEL parameter lists I/O paths at the level you specify:

- Enter 1 to display channel adapter information
- Enter 2 to display device adapter information
- Enter 3 to display device information

The MANAGER parameter lists the I/O paths associated with the given manager or managers, if the manager(s) exist. If not, IOCONFIG displays a warning message.

Use the DEST parameter to send LISTPATH output to the file IOCLIST. This file remains open until you exit IOCONFIG, at which point the file is closed and printed.

Deleting an I/O Path

The system manager or operator can delete an I/O path and all paths below it from the configuration with the DELETEPATH command. IOCONFIG will only delete a path if it is not currently in use or if it does not have an associated device. If either is true when you issue the DELETEPATH command, a warning message appears.

The syntax of the command is:

```
DELETEPATH [PATH =]device path
```

Currently, the only paths that can be completely deleted are those using a single-ended SCSI adapter.

Updating the device class association table

Many of the configuration commands implicitly update one or more system reserved device classes. For example, the ADEV command when used to configure a tape device, implicitly adds the new device in TAPE device class. Also, these commands explicitly update one or more device classes specified with the command.

The device class association information used by the ASSOCIATE and DISASSOCIATE commands is stored in ASOCIATE.PUB.SYS. This file is built when the system manager runs the program ASOCTBL.PUB.SYS. Therefore, after you have issued IOCONFIG commands, it is a good idea to update the file ASOCIATE.PUB.SYS by running ASOCTBL.

Using SYSGEN

A new command, `DOIONOW`, has been added to MPE/iX which allows you to make device configuration changes with SYSGEN and then immediately implement them. This means that system managers and operators can use the customary utility for device configuration rather than the new utility, and they need not reboot the system to change the configuration.

Using SYSGEN to modify the configuration online involves two steps:

1. The system manager modifies the SYSGEN's IO configuration to add or delete one or more devices as desired and then exits SYSGEN. These changes are saved in the file LOG4ONLN.PUB.SYS, which is used as input for the `DOIONOW` command (in the next step).
2. At the CI prompt, the system manager issues the new `DOIONOW` command. If the LOG4ONLN file is not found, you will see the following error message displayed on the console:

```
NO PENDING SYSGEN CONFIG CHANGES
```

The `DOIONOW` command creates a log file, ONLNOLOG.PUB.SYS, which is used to record the commands executed by IOCONFIG. If the `DOIONOW` command generates an error, you may view this same log file to determine the cause. This log file is purged, and a new one created, prior to invoking the new configuration which means that it only contains information for the last execution of IOCONFIG. Once you have found the problem, you can then edit the input file for the `DOIONOW` command, LOG4ONLN.PUB.SYS, to avoid duplicating the events that created the error.

MPE/iX Host Control for Tape Drives

by Larry Nichoalds
Commercial Systems Division

The DEVCTRL utility has been enhanced for the MPE/iX 5.5 Release. In addition to the previous features allowing for tape compression and eject settings, the 5.5 version of DEVCTRL also supports a new `LOAD` feature.

The `LOAD` option allows a tape device to be put online or taken offline without manual intervention. For example, this feature can be used as part of a job stream or run from a remote terminal to put a tape back online following a RESTORE operation eliminating the need to manually load the tape.

Note

If the eject option is enabled and an offline request is issued, the tape ejects, which requires manual intervention for further use of the tape. Similarly, if an online request is made following a RESTORE where the eject option had been enabled, a subsequent online request produces a `NOT READY` message at the console, again requiring manual intervention to reload the tape.

For data compression, disabling the data compression option allows for tapes to be created which can then be read by non-compressing DDS devices. Later on, you may enable compression mode again to create compressed tapes.

To reset options, you must run the utility again with different configuration options.

The following table defines how each of the different tape drives work with DEVCTRL:

Tape Drives	**Data Compression**	**Tape Eject**[1]	**Load**	**Note**
HP1504B/1521B	Yes	Yes	Yes	SCSI Data Compressing
HP1503B/1520B	No	Yes	Yes	SCSI Non Data Compressing
HP1502A/1512A	No	Yes	Yes	SCSI Non Data Compressing
HP1501A/1511A	No	Yes	Yes	HPIB Non Data Compressing
STK 4280/4220	Yes	No	No[2]	With HP 1.1 FW or later
HP7980S/HP7980SX	No	Yes	Yes	SCSI 1/2-inch tape
HP7980/HP7980XC	No	Yes	Yes	HPIB 1/2-inch tape, see Important Details note
Other HPIB Tape	NA	NA	NA	7978, 7976, 7974

1 All the tape drives do not maintain the eject settings when the system is rebooted.

2 The STK device does not support online requests from the host.

DEVCTRL can be run directly from the console, other terminal or incorporated into a job stream.

The following defines the command-like syntax for DEVCTRL:

Syntax

```
DEVCTRL  logical device [compression = [enable|disable|nochange]]
                        [eject = [enable|disable|nochange]]
                        [load = [online|offline|nochange]]
```

Examples

To enable data compression and the eject feature for LDEV 50:

```
DEVCTRL  50 compression=enable eject=enable
```

To enable the eject feature for only LDEV 50:

```
DEVCTRL 50 eject=enable
```

To put LDEV 50 on-line:

```
DEVCTRL 50 load=online
```

The logical device number is a required field, but compression, eject, and load are optional and can appear in any order. If the keywords are absent, the order is interpreted as compression first, eject second, and load third. For example,

```
DEVCTRL 50 enable disable online
```

would enable compression, disable the eject feature, and put the tape online.

Note

If you don't know the status for a tape drive (whether it's enabled or disabled), reset it.

Important Details Please Read

DEVCTRL and its companion program DEVTOOL, are located in the MPEXL.TELESUP group and account. If you need to relocate the utility, be certain to move only the DEVCTRL command file. DEVCTRL expects to find the DEVTOOL program in the MPEXL.TELESUP group and account. If DEVCTRL is to remain in MPEXL.TELESUP, you must either fully qualify the name (that is, DEVCTRL.MPEXL.TELESUP), or you must modify your HPPATH environmental variable.

For 7980 and 7980XC, the eject feature can be configured into the device through the front panel using configuration option 81.

If you use both compressing and non-compressing DDS tape drives, you may inadvertently put a compressed tape into a non-compressing drive. The results of this mistake vary depending on the type of non-compressing drive in which the tape is placed.

When using an HPC1503B, the following AVR message is created in response to a compressed tape mount:

```
I/O error ignored during AVR.  I/O status % 74.
```

For an HPC1300H (HPIB DDS) or an HPC1502A, no AVR error is created. However, any attempt to access the data on the tape results in an I/O error. For example, if you execute a RESTORE, the RESTORE aborts with a CIERR 1091.

New High Performance CD-ROM Drive for MPE/iX 5.5

by Jeff Flowers
Worldwide Customer Support Operations

Beginning with MPE/iX Release 5.5, the HP5401 CD-ROM drive will be available on the high-end systems HP 3000 Series 996/x00, 9x9KS/x00, 9x9KS/x20, 969/x00, 969/x20 and 9x8.

With a 4X data transfer rate (600KB/second), the high performance HP5401 CD-ROM drive is twice as fast as previous models, reducing system downtime needed for installing/upgrading MPE/iX software by half. Support costs are also reduced, owing to lower software distribution and manufacturing costs of CD-ROM media as compared to conventional tape and print media.

The HP5401 CD-ROM is industry standard and is compatible with all media currently available for CD-ROM readers. For further information on the HP5401 CD-ROM Drive, please contact your sales representative.

Configuration Information

The HP5401 CD-ROM has a configuration ID of CD-ROM-XM-5401TA, and is supported on the MPE/iX 5.5 Release.

New Optical Library for MPE/iX 5.5

by Jeff Flowers
Worldwide Customer Support Operations

With the MPE/iX 5.5 Release, the new HP optical library HPC1100B is now available for use with HP's TurboSTORE backup product. The HPC1100B has the same storage capacity and double density format as the HPC1708T/18T disk readers it replaces at a lower cost.

Features Include:

- 20 GB storage capacity
- 1 Double density disk reader (1.3 GB per media)
- 16 Media storage slots

The HPC1100B is fully compatible with HP's other drives and library products and complies with industry standards for magneto-optical disk interchange.

For ordering and configuration information, please contact your HP sales representative.

New Mass Storage Systems for MPE/iX 5.5

by Jeff Flowers
Worldwide Customer Support Operations

Introduction

Two new mass storage systems are now available from HP, utilizing state-of-the-art disk drives, redundant power supplies (if so configured), and DDS tape drives in a fault-resilient cabinet. The HP3311A (deskside) and HP3312A (rackmount) contain storage bays for devices that are hot pluggable, allowing removal and replacement without powering down the cabinet or the host (see the "Disk Replacement Example" section below).

Interface modules in the peripheral device pods for the cabinet make this possible by eliminating power and signal glitches that can occur when a device is removed or inserted from the cabinet. Depending upon the device and configuration, system software will recover communication links to the device through timeouts then reset sequences. Power supply and fan modules can be swapped provided there is another like module present. Disks can be swapped if the Mirrored Disk/iX product is in use for those disks. Hot add of disks is not supported.

Features

Following are the features of the two mass storage systems:

- Available in deskside and rackmount models
- Dual SCSI bus: either single-ended or fast-wide differential
- Up to eight 1-inch high or four 1.6-inch high storage modules per cabinet
- Two redundant fan modules (shipped standard)
- Two redundant power modules (one standard, one optional)
- Hot pluggable, front-access device modules

Benefits

- Modular design allows for configuration flexibility, as well as easy device installation/removal.
- Hot pluggable modules can be replaced while the host is operating. A bus reset is mechanically initiated when modules are removed or inserted. System device software will then attempt to reset I/O communication.
- Dual SCSI bus can be configured with two single-ended buses, two fast-wide differential buses, or one of each. An external cable can be used to combine two like buses into a single bus, if desired
- Modules easy to install/remove, yet lock in place to prevent accidental disconnection

Module Information

Fan

Fan modules are installed at the top of the cabinet. Although a single fan has sufficient capacity to cool the entire cabinet, it is necessary to have both fans present for proper airflow and to avoid problems should the remaining fan fail. The fan can be replaced while the cabinet is in use.

Power supply

Power supplies are installed at the bottom of the deskside unit (HP3311A) or at the side of the rackmount unit (HP3312A). As with the fan units, one is sufficient to power the entire cabinet, although for fault resilience, both power supplies are needed. Additional protection can be gained by using a separate UPS for each power supply. Another benefit of using the second power supply is the ability to replace a failed unit while the cabinet is in use.

DDS-2 Tape Drives

Tape drives require two 1-inch slots, and use the single-ended SCSI interface. If no disk drives share this interface, the DDS-2 drives are hot pluggable, otherwise they are not.

Disk Drives

Disk drives are available as either 2 or 4GB, single-ended or fast-wide differential SCSI. The 2 GB disks are one-inch form factor (low profile), the 4 GB disks are 1.6-inch form factor (half-height). The disks are hot pluggable in conjunction with the Mirrored Disk/iX product (see the example below)—removing or replacing a disk without this product is not supported.

Notes

The storage cabinet can accept up to eight 1-inch devices, four 1.6-inch devices, or a combination or 1 and 1.6 inch devices. This allows for a maximum of 16 GB in a cabinet that measures:

16.35-in(H) x 12.21-in(W) x 12.61-in(D)
in the deskside configuration

or

10.36-in(H) x 16.75-in (W) x 10.44-in (D)
in the rackmount configuration

Disk Replacement Example

An example of replacing a failed member of a mirrored pair of disks would be as follows:

1. Host notifies console operator that a mirrored disk has failed, placing partner disk in non-mirrored state:

   ```
   ?09:09/12/MIRRORED VOLUME DISABLED ON LDEV# 32
   ```

2. Operator reply requested to acknowledge this situation:

```
?09:09/22/ACKNOWLEDGE MIRRORED VOLUME DISABLED ON LDEV# 32 [Y/N]?
```

3. Operator replies to message:

   ```
   :REPLY 22,Y
   ```

4. Pull out lever on disk module and remove module from cabinet.
5. Slide new disk module into slot (of same size and type as old disk).
6. Run VOLUTIL to repair the mirrored volume:

   ```
   volutil: REPLACEMIRRVOL PROD_SET:MEMBER2 32
   ```

   The system now recognizes (mounts) the replaced volume, resumes disk mirroring, and starts the repair process. The replacement volume has the same characteristics specified when the disabled volume was first initialized using the NEWMIRRVOL or NEWMIRRSET commands.

Further Information

For further information, please refer to the *HP A3311A/HP A3312A Storage Enclosure User's Guide* (A3311-90002) or contact your HP sales representative.

New Disk Drives for MPE/iX 5.5

by Jeff Flowers
Worldwide Customer Support Operations

MPE/iX 5.5 (or MPE/iX-Express 3 based on General Release 5.0), introduces support for a series of new disk drives. These disks drives offer higher capacity with higher performance than previous drives. To take full advantage of system powerfail recovery, an HP PowerTrust UPS is needed. For information on the HP PowerTrust UPS and system powerfail recovery, see the technical articles, "UPS Required for Support of New SCSI Disks" and "Protective System Abort #5300 From UPS Monitor/iX" in Chapter 10, "Technical Articles."

The two tables below list each disk drive's Product Description, configuration IDs, and ordering part numbers. For further information on these or other HP products, please contact your HP sales representative.

External Disk "Classic" Cabinet (HP C3022T/C3023T/C3024T)

Product Description	Disk Configuration ID	SPU Integrated	Field Integrated	Deskside	Upgrade Kit
1x1GB SE Low Profile	ST31200N ST31230N	A3349A			
1x2GB SE Low Profile	ST32550N VP3215S	A3304A	C5254RZ	C5254R	C5254T C5257U
2x2GB SE Low Profile	ST32550N VP3215S		C5255RZ	C5255R	C5255T
3x2GB SE Low Profile	ST32550N VP3215S		C5256RZ		
1x2GB FW Low Profile	ST32550W VP3215SW	A3351A		C5258R	C5258T C5261U
2x2GB FW Low Profile	ST32550W VP3215SW		C5259RZ	C5259R	C5259T
5x2GB FW Low Profile	ST32550W VP3215SW		C5260RZ		
1x4GB SE Half Height	ST15150W	A3352A	C5262RZ	C5262R	C5262T C5263U
1x4GB FW Half Height	ST15150W	A3353A	C5264RZ	C5264R	C5264T C5266U
5x4GB FW Half Height	ST15150W		C5265RZ		

External Disk High Availability (HASS) Cabinet (HP A3311A/A3312A)

Product Description	Disk Configuration ID	Cabinet Integrated	Field Integrated	Deskside	Upgrade Kit
Enclosure		A3312AZ	A3312A	A3311A	
1x2GB SE Low Profile	ST32550N VP3215S	Opt. 121	Opt. 121	Opt. 121	A3317A
2x2GB SE Low Profile	ST32550N VP3215S	Opt. 122	Opt. 122	Opt. 122	
1x2GB FW Low Profile	ST32550W VP3215SW	Opt. 123	Opt. 123	Opt. 123	A3318A
2x2GB FW Low Profile	ST32550W VP3215SW	Opt. 141	Opt. 141	Opt. 141	A3319A
1x4GB FW Half Height	ST15150W	Opt. 143	Opt. 143	Opt. 143	A3320A

MPE/iX Now Supports 3.75GB of System Memory

by Ed Olander
Commercial Systems Division

MPE/iX Release 5.5 now supports 3.75GB of main memory. Actually, a full address space of 32-bits is now usable by both the operating system and the physical hardware. A full 32-bit address space supports up to 4,294,967,296 bytes of physical memory. We shorten the preceding statement by saying a 32-bit address space provides 4GB of memory.

If a 32-bit address space provides 4GB of memory why do we use the expression—3.75GB of main memory? The answer comes from the architected definition of physical memory as seen by the HP PA-RISC Computer Architecture. The physical address space for PA-RISC machines has three components:

- **Memory Address Space**

  This address space is called the main memory. This space represents 15/16ths of the physical address space.

- **PDC Address Space**

  This address space is used to reference Processor Dependent Code (PDC) and its associated resources. This space represents 1/256ths of the physical address space.

- **I/O Address Space**

  This address space is used to reference I/O registers. This space represents 15/256ths of the physical address space, or the PDC and I/O address space represents 1/16th of the physical address space.

Therefore, the maximum main memory for an address space of 32 bits is 3.75GB (15/16 * 4GB) for HP PA-RISC hardware platforms. With the release of the MPE/iX 5.5 operating system, MPE/iX is now completely compatible with a fully utilized 32-bit address space.

The HP 3000 996/995/991 Servers support 3.75GB of main memory.

1600 BPI Software Distribution

by Paul Nugent
Commercial Systems Division

Hewlett-Packard stopped distributing 1600 BPI tapes and eliminated 1600 BPI as a boot/update device on all HP 3000 MPE/iX products on September 30, 1995. 1600 BPI tape has technical limitations that prevent HP from adding new MPE/iX functionality, such as new devices and additional diagnostics. 1600 BPI tape drives can still be used for store/restore.

We believe the MPE/iX customers impacted are those who have only 1600 BPI tape drives; specifically, models HP7974A, HP7979A and HP7979S. If you have no update or installation device other than 1600 BPI, you will need to upgrade your hardware after MPE/iX General Release 5.0.

If you already have an update or installation device such as 6250 BPI, DDS, or CD-ROM, you simply need to update your support contract to reflect one of these formats for future MPE/iX software distributions. Please contact your service contract administrator to change your software distribution media.

10

Technical Articles

A Detailed Look at CI Enhancements

by Jeff Vance
Commercial Systems Division

Overview

This article provides a more indepth understanding of the following CI enhancements introduced in the "CI Enhancements Overview" article in Chapter 3, "System Management." Note that the CI enhancements in Release 5.5 are a superset of the CI enhancements in Express 3.

Following is a summary of the CI enhancements:

- Eight new CI variables were added: `HPREMIPADDR`, `HPREMPORT`, `HPLOCIPADDR`, `HPLOCPORT`,. `HPSTREAMEDBY`, `HPLASTJOB`, `HPOSVERSION`, and `HPRELVERSION`.
- CI string variables are now 1024 characters long. Additionally, approximately twice the number of user-defined variables can be created in 5.5.
- Five new CI evaluator functions related to text parsing were added: **`word()`**, **`edit()`**, **`repl()`**, **`delimpos()`** and **`pmatch()`**. Also the **`input()`**, **`str()`** and **`rht()`** functions were enhanced.
- Command files and UDCs can be protected by granting only execute (X) access to their users.
- Command files can use the full POSIX naming standards, and the `HPPATH` variable now supports directory names.
- The `REDO` command can upshift and downshift characters and also delete, upshift or downshift a word.
- The CI recognizes a leading "#" as a comment line.
- Online help is available for all CI variables and evaluator functions.

New CI Variables

Eight new predefined CI variables were added in MPE/iX Release 5.5:

```
HPLOCIPADDR
HPREMPORT
HPLOCIPADDR
HPLOCPORT
HPSTREAMEDBY
HPLASTJOB
HPOSVERSION
HPRELVERSION
```

HPREMIPADDR

`HPREMIPADDR` is a string variable that contains the IP address of a remotely connected user (client). If you are directly connected to the HP 3000, `HPREMIPADDR` is set to "" (empty string).

HPREMPORT

`HPREMPORT` is an integer variable that contains the TCP port number allocated on the remote machine (client) for use on the incoming TCP connection. If you are directly connected to the HP 3000, `HPREMPORT` is zero.

HPLOCIPADDR

`HPLOCIPADDR` is a string variable that contains the IP address of the HP 3000 local Network Interface (NI) that will be used for outbound data. If you are directly connected to the HP 3000, `HPLOCIPADDR` is set to "". HP 3000s support several network interface cards and thus can have more than one IP address.

Note

`HPLOCIPADDR` may not reflect the IP address of the NI used for inbound data flow from the same client due to either network load balancing or a misconfiguration.

HPLOCPORT

`HPLOCPORT` is an integer variable that contains the local TCP port number for the network service provided to the client. If you are directly connected to the HP 3000, `HPLOCPORT` is zero. For example, a telnet connection uses port 23, NS/VT connections use ports 1537 and 1570, an ftp data connection uses port 20.

Below is a sample logon UDC that filters logons based on time-of-day, inbound IP addresses and inbound port number (service being used). Security on this UDC is assigned so that only the file's creator has read/write access, all others have only execute access.

```
SYSLOGON
OPTION LOGON, NOBREAK
# This system-wide logon UDC ensures that no one logs on "after hours"
# from outside our company via NS/VT.

# After hours are weekends and weekdays after 6pm and before 7am
setvar too_early 7
setvar too_late  6

# Currently only checking for VT connections, later will add telnet check
setvar msg_mode_VT    1537
setvar stream_mode_VT 1570

# Check weekday and time of day first
if word(hpdatef) = "SAT" or word(hpdatef) = "SUN" or &
   (word(hptimef,,-1) = "PM" and ![word(hptimef,":")] > too_late)  or &
   (word(hptimef,,-1) = "AM" and ![word(hptimef,":")] < too_early) then

   # It is too late or too early or the weekend.
   # Check for an offsite IP address via NS/VT.
   if HPREMIPADDR <> "" then
      # A network connection
      # Determine n/w mask based on host's IP address class
      setvar class_octet ![word(HPLOCIPADDR,".")]

      if class_octet < 128 then
         # Class A addr
         setvar nw_mask word(HPLOCIPADDR,".")
      elseif class_octet < 192 then
         # Class B addr
         setvar nw_mask lft(HPLOCIPADDR,pos(".",HPLOCIPADDR,2)-1)
      else
         # Assume Class C addr
         setvar nw_mask lft(HPLOCIPADDR,pos(".",HPLOCIPADDR,3)-1)
      endif

      # See if connection is outside the company
      if lft(HPREMIPADDR,len(nw_mask)) <> nw_mask then
         # Outside connection, check for NS/VT
         if HPLOCPORT = msg_mode_VT or HPLOCPORT = stream_mode_VT then
            echo Connection refused, please contact MIS.
            bye
         endif
      endif
      deletevar class_octet, nw_mask
   endif
endif
deletevar too_early, too_late, msg_mode_VT, stream_mode_VT
echo Welcome to !HPSYSNAME.
**********
```

HPSTREAMEDBY

HPSTREAMEDBY is a read-only string variable. Typically, it contains the user.account name of the user who streamed a job, or who STARTSESSed a session. The exact format is:

`UserName.AcctName (#J|Snnnnn)`

However, for the initial OPERATOR.SYS logon or a job streamed from the SYSSTART.PUB.SYS file, the job/session ID is replaced by the string, SYSTEM PROCESS. For example, MANAGER.SYS (SYSTEM PROCESS).

HPLASTJOB

`HPLASTJOB` is a read-write string variable. It contains the job ID of the job that was most recently `STREAM`ed. The format is: `#Jnnnnn` and appears the same as the job ID output by the `STREAM` command.

HPOSVERSION

`HPOSVERSION` is a read-only string variable. It contains the operating system version ID identical to the middle version string in the SHOWME banner. Remember that the `HPVERSION` variable contains the user version ID, which can be modified via SYSGEN.

HPRELVERSION

`HPRELVERSION` is a read-only string variable. It contains the Release version ID identical to the left version string in the SHOWME banner.

Longer CI Variables

In MPE/iX Releases 5.0 and earlier, CI variable values were limited to 255 characters. Beginning with MPE/iX Release 5.5, CI variable values can be up to 1024 characters long. Variable names can continue to be up to 255 characters long. Also the CI's variable table can expand to accommodate approximately two times more variables than previously.

In MPE/iX Release 5.5, the maximum number of variables that can be created is approximately 10700. This number is inversely related to the length of the variable's name and the length of its value. Also, the maximum number of variables that can be created with 254-character names and 255-character values is 2190, compared to 974 in MPE/iX Release 5.0.

New Evaluator Functions

Five new CI evaluator functions were added and three functions were expanded in MPE/iX Release 5.5. Most of these enhancements make parsing text easier via the CI.

The new CI evaluator functions are:

```
word()
edit()
repl()
delimpos()
pmatch()
```

The enhanced functions are:

```
str()
rht()
input()
```

word()

A general word extraction. String function.

Syntax: `word(`*str*`[,`*delims*`][,`*nth*`][,`*end_var*`][,`*start*`])`

This new function extracts the *nth* word from *str* beginning at *start*, delimited by one of the characters in *delims*, placing the index of the delimiter that terminated the word in a CI variable named by the *end_var* argument.

Parameters:

str	Required. *str* is any quoted string or string variable. This is the source for the word extraction.
delims	Optional. *delims* is a string that contains a list of characters that constitute the termination of a word. The default *delims* are: space, comma, semicolon, tab, equal sign, square brackets, single and double quotes, parentheses. All other characters in *str* are considered part of a word.
nth	Optional. An integer indicating which word to extract from *str*. The default value is 1, meaning parse out the first word, scanning from left to right. A value of 2 causes the second word in *str* to be extracted. A negative value means extract a word starting at the end of *str*, parsing from right to left. A value of -3 means to extract the third from last word in *str*. A value of -1 indicates the last word in *str*.
end_var	Optional. The actual name of a variable to be set to the index in *str* of the delimiter that terminated the extracted word. *end_var* cannot be a string expression— the actual unquoted name must be used. The default is to not set a variable. If the *nth* word is not found then *end_var* is not set. For *nth* >= 0 the largest *end_var* value is `len(str)+1`, meaning the last word was found. For *nth* < 0 the smallest *end_var* value is 0, meaning the first word was found. Multiple spaces in *str* after the extracted word are skipped.
start	Optional. An integer index into *str* where the word extraction begins. The default *start* when *nth* >= 0 is 1, and when *nth* < 0 is the index of the last byte in *str*.

`word()` starts at *str[start]* and scans *str* looking for a word delimiter. The direction of the scan is determined by *nth*. Positive *nth* values cause a left to right scan, whereas negative values use a right to left scan.

Initial spaces are skipped. After the delimiter is found, trailing spaces are also skipped until a non-space delimiter is found, if any. *end_var* is set to the index of this non-space delimiter. If one or more spaces are the only delimiter between words then *end_var* is set to the index of the last space in *str* before the next word.

There are two cases when `word()` can functionally return an empty string:

1. When the *nth* word does not exist in *str*.
2. When the *nth* word exists but has no value. Consider:

   ```
   setvar str, "rec=10,,f"
   ```

   - `word(str,,5)` returns `""`. since there are only four words in *str*.
   - `word(str,,3)` returns `""`. since the third word has no value—the third word starts at *str[8]* and ends at the same index.

   These two cases can be distinguished by passing the *end_var* parameter.

   - `word(str,,5,j)` does not set the variable J.
   - `word(str,,3,j)` sets J to 8.

   Examples:

   ```
   word('file a=bb,old;rec=40,,f,ascii')             = 'file'
   word('file a=bb,old;rec=40,,f,ascii',,2)          = 'a'
   word('file a=bb,old;rec=40,,f,ascii',";,",,j,8) = 'bb', j=10
   word('file a=bb,old;rec=40,,f,ascii',,-4,j)       = '40', j=18
   ```

   Here's how to parse every token in a string by incrementing the starting index.

   ```
   setvar j 0
   while setvar(j,j+1) <= len(str) do
      setvar token word(str,,,j,j)
      ...
   endwhile
   ```

   Another way to parse a string by using *nth* and omitting *start*:

   ```
   setvar j 0
   setvar cnt 0
   while setvar(cnt,cnt+1) <= 9999 and j <= len(str) do
      setvar token word(str,,cnt,j)
      ...
   endwhile
   ```

edit()

Full REDO-like editing of a string. String function.

Syntax: `edit(`*str*`,`*editstr*`[,`*start*`])`

This new function applies the edit contained in *editstr* to *str* starting at *start* and functionally returns the result.

Parameters:

str	Required. *str* is any quoted string or string variable. The edit is applied to *str*.

editstr	Required. *editstr* is a string containing a REDO-like edit. For example, "dddiXYZZY" would delete the 2nd, 3rd, and 4th characters in *str* and then insert "XYZZY".
start	Optional. An integer index into *str* where *editstr* is applied. The default *start* is 1.

`edit()` starts at *str[start]* and applies the edit in *editstr* exactly as the CI's REDO command edits a command line. The edited string is functionally returned. The size of the string that can be edited is limited to the size of the CI's command buffer, which is currently 512 bytes.

Examples:

```
edit('abcdefg','>dd')          = 'abce'
edit('ab cd;g','dwd')          = 'cd;g'
edit('abccd;g','c/c/AA/')      = 'abAAAAd;g'
edit('abcdefg','^^',3)         = 'abCDefg'
```

repl()

General string replacement. String function.

Syntax: `repl(`*str*`,`*oldstr*`,`*newstr*`[,`*cnt*`][,`*start*`])`

This new function replaces *cnt* occurrences of *oldstr* in *str* with *newstr*, starting at *start*.

Parameters:

str	Required. *str* is any quoted string or string variable. This is the source for the replacement.
oldstr	Required. *oldstr* is the string searched for in *str*. It is replaced by *newstr*. No replacement occurs if *oldstr* is empty.
newstr	Required. *newstr* is the string that is substituted for *oldstr* and it can be empty ("").
cnt	Optional. *cnt* is the number of replacements to be done in *str*. The default *cnt* is zero meaning replace all occurrences of *oldstr* in str. A positive *cnt* indicates that the replacements are done from left to right, e.g., 1 means replace the first occurrence, 2 means the first two occurrences, etc. A negative *cnt* indicates right to left replacement, e.g., -1 means replace the last occurrence, -2 means the last two occurrences, etc.
start	Optional. An integer index into *str* where the replacement begins. The default *start* when *cnt* $>=$ 0 is 1, and when *cnt* $<$ 0 is the index of the last byte in str.

It is possible for the replace operation to overflow *str*. In that case the maximum number of replacements are done prior to exceeding the maximum size of a string variable (currently 1024 bytes).

Examples:

```
repl('aaabcaab','aa','X')      = 'XabcXb'
repl('aaabcaab','ab','',-1)    = 'aaabca'
repl('f*.*','*','@')           = 'f@.@'
```

delimpos()

Finds the position (index) of a delimiter. Integer function.

Syntax: `delimpos(`*str*`[,`*delims*`][,`*nth*`][,`*start*`])`

This new function returns the position of the *nth* delimiter in *str* beginning at *start*.

Parameters:

str	Required. *str* is any quoted string or string variable. This is the source for the delimiter searching.
delims	Optional. *delims* is a string that contains a list of one or more characters that are searched for in *str*. The default *delims* are the same as for `word()`: space, comma, semicolon, tab, equal sign, square brackets, single and double quotes, parentheses.
nth	Optional. An integer indicating which delimiter occurrence to scan for. The default value is 1, meaning find the position of the 1st delimiter in *str*. A value of 2 means find the second delimiter, etc. A negative value means search for the delimiter from right to left. A value of -3 means to find the third from last delimiter in *str*. A value of -1 indicates find the index of the last delimiter in *str*.
start	Optional. An integer index into *str* where the delimiter searching begins. The default *start* when nth >= 0 is 1, and when *nth* < 0 is the index of the last byte in *str*.

There are several differences between `delimpos()` and `pos()`. The `pos()` function supports the matching of one or more characters, e.g., `pos('abcd',str,2)` locates the index in *str* of the 2nd occurrence of "abcd". The `delimpos()` function only supports single character matches. However, several characters can be tested for a match. For example, `delimpos(str,'abcd',2)` locates the index in *str* of the second occurrence of **either** an "a" **or** "b" **or** "c" **or** "d".

Typically, the delimiter string will be token separators like semicolon (;), comma (,), double quotes (" "), etc. The `delimpos()` function also supports the start parameter which is not available in `pos()`.

Examples:

```
                       1         2                         Funct  Delim
              12345678901234567890123456789                 Rtn   Found
delimpos('file a=bb,old;rec=40,,f,ascii')                   5     ' '
delimpos('file a=bb,old;rec=40,,f,ascii',,3)                10    ','
delimpos('file a=bb,old;rec=40,,f,ascii',",;",-4)           14    ';'
delimpos('file a=bb,old;rec=40,,f,ascii',,,7)               7     '='
```

pmatch()

Pattern matching. Boolean function.

Syntax: `pmatch(`*pattern*`,`*str*`[,`*start*`])`

This new function returns TRUE if *pattern* is found in *str*, starting at *start*.

Parameters:

pattern	Required. A string pattern to be matched in *str*. Wildcard characters are supported.
str	Required. *str* is any quoted string or string variable. This is the source that the pattern is matched against.
start	Optional. An integer index into *str* where the search for the pattern begins. The default *start* is 1.

str[start] is scanned looking for pattern. If a match is found then TRUE is returned, else FALSE is returned. The pattern can be any MPE filename character, including all wildcards. `pmatch()` provides the same pattern matching used by `LISTFILE` and `SHOWVAR`.

`pmatch()` is different from `pos()` since wildcards are supported, a *start* parameter is provided, which occurrence of *pattern* to match cannot be specified, and true pattern matching is performed. For example, `pos('abc','aaabccc')` is 3. `pmatch('abc','aaabccc')` is FALSE. However, `pmatch('@abc@','aaabccc')` is TRUE.

Both *pattern* and *str* are limited to 256 bytes due to internal interface restrictions. If *pattern* or *str* are empty (but not both) then FALSE is returned. If both *pattern* and *str* are empty then TRUE is returned.

Examples:

```
pmatch('f@','fread')     = true
pmatch('f#','fabc')      = false
pmatch('@f@,'abcdefg')   = true
pmatch('abc','abcd')     = false
pmatch('','abc')         = false
```

str()

General string extraction. String function.

Syntax: `str(`*str1*`,`*start*`,`*cnt*`)`

This existing function extracts *cnt* characters from *str1* starting at *start*.

The *cnt* parameter was enhanced to support an ending index rather than only a byte count. If *cnt* is negative then the absolute value of *cnt* is the ending index in *str1* to stop the extraction. If *cnt* is negative and -*cnt* is less than *start* then an empty string is returned.

Examples:

```
str('abcde',2,3)      = 'bcd'
str('abcde',2,-4)     = 'bcd'
str('abcde',3,-3)     = 'c'
str('abcde',3,-1)     = ''
```

rht()

Right-hand string extraction. String function.

Syntax: `rht(`*str1*`,`*cnt*`)`

This existing function extracts the right-most *cnt* characters from *str1*.

The *cnt* parameter was modified to support a starting index rather than only a byte count. If *cnt* is negative then the absolute value of *cnt* is the starting index in *str1* to start the extraction, which includes all characters from *str1[-cnt]* to the end of *str1*.

Examples:

```
rht('abcde',3)        = 'cde'
rht('abcde',-4)       = 'de'
rht('abcde',-1)       = 'abcde'
```

input()

Read from $STDIN. String function.

Syntax: `input([`*prompt*`][,`*wait*`][,`*cnt*`])`

This existing function optionally writes *prompt* to $STDLIST, reads from $STDIN, with the option of the read being a timed read of *wait* seconds. The input from $STDIN is functionally returned.

The *cnt* parameter is new. If *cnt* is specified, then only *cnt* bytes are read from $STDIN. The default *cnt* is the maximum size of a string variable, currently 1024 characters. If *cnt* is specified and less than *cnt* characters are supplied as input, you must still use the RETURN key to send the data.

Example:

```
input('Do this (Y/n)?',10,1)
```

Prompts to $STDLIST, does a 10 second timed read on $STDIN of 1 character.

Execute Access for UDCs and Command Files

Prior to the MPE/iX-Express 3 Based on General Release 5.0. all UDC and command file users must be allowed read access to the UDC file or the command file they wish to execute. Although these user command files can be protected by lockwords. the user must know the lockword to be able to read the file's contents.

Now UDC files and command files can be protected by denying READ (R) access and granting EXECUTE (X) access to users that need to execute the file but are not permitted to read the file. For example. using either of the following command lines grants execute access to the `mycmdf` file. You can verify the security using `LISTFILE` formats -2 or 4.

```
:altsec mycmdf; access=(x:any; r,w,l,a:gu)
```

or

```
:altsec mycmdf; repacd=(racd,x:@.@; r,w,l,a:$group)
```

If you lack READ access to a command file or UDC file. the system behaves in the following manner:

- You cannot see any of the commands within the file. Specifically. `OPTION LIST` and the `HPCMDTRACE` variable are defeated.
- `HELP` is unavailable for the file. For a UDC file this means that all of the UDCs within the file are treated as if `OPTION NOHELP` was specified.
- `SHOWCATALOG` still lists the individual UDCs and UDC filenames.
- If an error occurs, the offending command line is not echoed to $STDLIST.

Of course, if you have READ access to the file then everything works in a compatible manner.

POSIX-Named Command Files

In MPE/iX General Release 5.0. all command files must follow MPE naming rules and the `HPPATH` variable can only contain the names of MPE groups and accounts.

In the Express 3 Release, command files can reside in the Hierarchical File System (HFS) and follow the more flexible POSIX naming conventions. For example, a command file can now be named find_deduction, 123, or AutoExec.BAT. UDC files are still restricted to MPE naming rules.

Qualified MPE or POSIX filenames are executed immediately, skipping the `HPPATH` variable. For example, `file.grp`, `*feq`, `$oldpass`, `/bin/ls`, `./do_it` are all qualified filenames and thus `HPPATH` is ignored. If the file exists, it is executed. If the file is not found, then the following message is reported:

```
Unknown command name. (CIERR 975)
```

Qualified MPE filenames are `file.group` or `file.group.account`. Also, back references to a file equation and system-defined files, such as $OLDPASS, are considered qualified MPE names. Qualified POSIX names are absolute pathnames (the name starts at root), or Current Working Directory (CWD) relative names (the name starts at the user's current working directory, `./name`).

The command file named `a/b` can be considered a qualified POSIX name (file `b` under directory `a`), but for compatibility reasons this is first treated as an unqualified MPE name with a supplied lockword. Actually, `file.grp` could be an unqualified POSIX name (in which case `HPPATH` is used), but the MPE file, `FILE`, in the MPE group, `GRP`, is looked for first. Other examples where the command filename could be both an MPE name or a POSIX name are covered later, but in all of these cases the MPE name is searched for before the POSIX name.

To execute unqualified POSIX-named command files, `HPPATH` must contain one or more entries specified in MPE-ESCAPED syntax. That is, the name must begin with a dot (".") or a slash ("/"). The default `HPPATH` setting is `!hpgroup,PUB,PUB.SYS,ARPA.SYS`. Since the default `HPPATH` contains no MPE-ESCAPED named entries, unqualified POSIX-named command files cannot be located.

If `HPPATH` is modified to be `PUB.SYS,/bin,./mybin`, then unqualified POSIX-named command files can be located in `/bin` and in `CWD/mybin`. If the System Manager desires to place POSIX-named command files in PUB.SYS then `HPPATH` needs to contain a `/SYS/PUB` entry. In other words, to use unqualified POSIX-named scripts, even if the file resides in an MPE group, the location (directory) name must appear in `HPPATH` in MPE-ESCAPED syntax.

Following is the basic algorithm for processing MPE- or POSIX-named command files:

- The command name is parsed twice: first via MPE rules then via POSIX syntax rules.
- If the name is a qualified MPE name (e.g., `a.b`) it is tried first.
- If a file matching the name has not yet been found and the name is not a qualified POSIX name then `HPPATH` is used to try to locate the name.
    - MPE path elements (group.accounts) are appended to the MPE parsed version of the command filename.
    - POSIX path elements (directories) are prepended to the POSIX parsed version of the command name.
    - The name, qualified by the appropriate HPPATH element, is searched for until the first match.
- If a file matching the command filename has still not been located and the command name is a qualified POSIX name (e.g., `./a`, `/a`, `a/b`) then that exact name is searched for.
- At this point either there is a match or an `unknown command` error is reported.

Note

The same command filename can be both a qualified MPE name and a qualified POSIX name, e.g., `a/b.c`. This name could refer to file `A` in group `C` with lockword `B`, or file `b.c` under directory `a`. According to the above algorithm, the qualified MPE version of the name is tried first.

Before showing some examples it is important to remember that MPE command names are delimited by the first character that is not a valid name character. This is how the CI has behaved since MPE XL Release 1.0 and, in most cases, a blank delimits all command names. UDC names are delimited by the first non-alphanumeric character. For example, if the command entered is `:udc1a.chv`, the CI first looks for a UDC named `UDC1A`, and if found, passes `.chv` as the first argument.

Built-in command names are delimited by the first non-alpha character. For example, if the built-in command is `:run$oldpass`, `$oldpass` is passed as the first parameter to the `RUN` command. Likewise, if the built-in command is `:abortio7`, 7 is passed as the first argument to the `ABORTIO` command.

Command filenames are delimited by the first non-filename character. For example, if the command entered is `:xyzzy.g%foo`, the file `XYZZY.G` is passed `%foo` as its first argument. Again, this is not new behavior for the Express 3 Release, but it is worth describing since users may assume a blank delimiter is required.

For the examples below, assume that `HPPATH` is set as:

```
:setvar hppath "PUB.SYS, ., /SYS/PUB"
```

where the directory "." refers to the user's current working directory.

Command	**HPPATH Searching**
`:a`	A.PUB.SYS, ./a, /SYS/PUB/a
`:A`	A.PUB.SYS, ./A, /SYS/PUB/A (redundant)
`:./a`	Qualified name, `HPPATH` is ignored. ./a is executed
`:/a`	Qualified name, `HPPATH` is ignored. /a is executed
`:a_b`	A.PUB.SYS ("_b" passed as 1st parm), ./a_b, /SYS/PUB/a_b
`:a.b`	Qualified MPE name, A.B.ACCT, if not found then `HPPATH` is used as: ./a.b, /SYS/PUB/a.b (MPE path elements are skipped)
`:_a`	./_a, /SYS/PUB/_a (MPE path elements are skipped)

`:a/b`	A/B.PUB.SYS (POSIX path elements are skipped). if not found then qualified POSIX name a/b is executed
`:a/b.c`	Qualified MPE name, A/B.C, if not found then qualified POSIX name, a/b.c (`HPPATH` skipped)
`:_a/b`	Qualified POSIX name, _a/b (`HPPATH` skipped)

REDO Enhancements

The `REDO` procedure which is invoked by the CI, Debugger, SYSGEN, VOLUTIL and the Link Editor has been enhanced.

Motivated, in part, by the need for case specificity in POSIX filenames, `REDO` now supports upshift (`^`) and downshift (`v`) edits. Also, as filenames get longer, it becomes more cumbersome to delete a name. Therefore, a word edit has been defined. For example **`dw`** deletes a word. `REDO` defines words to be any characters delimited by a space, comma, semicolon, tab, equal sign, square brackets, single and double quotes, and parentheses. Words can also be upshifted (**`^w`**) or downshifted (**`vw`**).

Finally, sometimes the edit needs to be applied to part of a word or to several words, so you can supply your own matching delimiter. For example, **`d/`** would delete to the first "/".

The new edits can be applied to the end-of-line by preceding the edit with ">". For example, **`>dw`** deletes the last word, and **`>v@`** downshifts from the end-of-line to the first "@" character (searching from right-to-left).

The user-supplied delimiter edits are not performed if the delimiter is not found. Also the special characters space, "^" and ">" cannot be used as delimiters since these symbols have already been defined. If a word edit is specified and a word delimiter is not found then the edit applies to the entire line.

Online HELP for CI Variables and Functions

Information about all CI variables has been supported via `HELP` since MPE/iX Release 5.0, but Release 5.5 also includes HELP for all CI evaluator functions.

For example:

`HELP FUNCTIONS`	- Shows a table of all evaluator functions.
`HELP FINFO`	- Shows the details of the `FINFO` function.
`HELP WORD`	- Describes the new `WORD` function.
`HELP VARIABLES`	- Shows a table of all CI variables.
`HELP HPREMIPADDR`	- Describes the new `HPREMIPADDR` variable.

MPE/iX Dependent Libraries Technical Overview

by Huong Ho and Ed Olander
Commercial Systems Division

Intended Audience

This article is intended for those who would like a more indepth understanding of Dependent Libraries on MPE/iX Release 5.5. The readers may include: application developers, porters, support, and training personnel. For those who would like an introduction to this new functionality, please read the *Communicator* Overview articles, "Introducing Dependent Libraries on MPE/iX Loader" and "HP Link Editor/iX Enhancement," in Chapter 6, "Application Development."

Introduction

Currently on MPE/iX you can specify the libraries on which the program is dependent when creating an executable program file. This capability exists through the specification of the parameter `XL=` "string of *filelist*" of the `RUN`, `LINK`, and `ALTPROG` commands. The string *filelist* specifies the names of executable libraries that need to be searched to resolve external symbol references from the executable program. The created program is said to have a *dependency* on the specified libraries.

Beginning with MPE/iX Release 5.5, the above concept has been extended to the libraries themselves. It is now possible to specify a filelist for a library when it's built or it can be altered later. Since the library has a dependency on the specified libraries, it now becomes a Dependent Library. In addition to the Dependent Library feature, the loader procedure `HPGETPROCPLABEL()`, has been modified to handle data symbols as well as code symbols. Furthermore, `HPGETPROCPLABEL()` allows dynamically loaded libraries to bind their code symbols through already loaded libraries.

Important Details Please Read

In addition to the enhancements of `HPGETPROCPLABEL()`, the casesensitive parameter has been modified so that it works properly. Prior to MPE/iX Release 5.5, `HPGETPROCPLABEL` allowed calls to be casesensitive, even if casesensitive was set to FALSE.

This casesensitive modification may cause previously running applications (specifically: third-party programs), to fail with a Data Memory Protection Trap (DMPT). This is due to naming conflicts for intrinsics with similar naming conventions such as FREAD and fread(). Please refer to the "casesensitive" description in the "HPGETPROCPLABEL()" section in this article for more details.

All these capabilities will bring MPE/iX executable libraries even closer to HP-UX shared libraries and make porting products from HP-UX to MPE/iX even simpler.

Note

Conceptually, MPE/iX Dependent Libraries are analogous to HP-UX Shared library dependencies. However, due to compatibility requirements in implementation, MPE/iX and HP-UX have different interpretations of HP-UX load graphs and MPE/iX *provisional binding sequences* used to discard duplicates. Please refer to the last section for more details.

This article discusses what a developer of MPE/iX needs to know to use the Dependent Libraries feature and the enhanced `HPGETPROCPLABEL()` capabilities. The sections of this article are summarized as follows:

- **How to Use Dependent Libraries from a Developer's Perspective**

  This section discusses the enhancement to:

  - Link Editor
  - `RUN` command semantic
  - `HPGETPROCPLABEL()`

- **Extensions to the MPE/iX binding sequence**

  MPE/iX has the concept of a binding sequence which pertains to the temporal order in which libraries are opened and searched.

  The MPE/iX loader currently supports two basic binding sequences (i.e., an ordered list of libraries): a static binding sequence and a dynamic binding sequence. With the addition of Dependent Libraries, MPE/iX can now support more complex binding sequences. This section discusses new capabilities in controlling these binding sequences. Key topics discussed are the following:

  - Dependency Lists
  - Dependency Tree
  - Binding Tree
  - Provisional Binding Sequence
  - Position of Dynamic Libraries
  - Positional Dynamic Loading

- **Variance of MPE/iX Dependent Libraries and HP-UX Shared Library Dependencies**

  Although there has been every intention to make the new capabilities of MPE executable libraries close to the HP-UX shared libraries, there are some differences. This section illustrates the differences.

How to Use Dependent Libraries from a Developer's Perspective

From a developer's perspective, Dependent Libraries enable more libraries to be dynamically loaded in addition to those that are currently loaded. Basically, the loader loads an executable library and then loads the libraries it depends on if they exist and are not previously loaded. The necessary components that allow accesses to Dependent Libraries are:

- Link Editor.
- `RUN` command.
- Dynamic loading via `HPGETPROCPLABEL()`.

Link Editor

The MPE/iX Link Editor has been enhanced to support Dependent Libraries. The Link Editor command `BUILDXL` has a new parameter `lib=` that is used to specify the dependency list of libraries. And, a new Link Editor command `ALTXL` has been added to enable the dependency list of libraries to be altered.

For more information regarding the Link Editor changes, please read the *Communicator* technical article, "HP Link Editor/iX Enhancements Detail," in this chapter.

RUN Command Semantics

The semantics of the CI `RUN` command are different in one sense. Until now, only the program file could have a dependency list of libraries. The Link Editor would check to see if there were multiple specified libraries and fail the link if there were. This is no longer the case. Libraries can have a dependency list too. The Link Editor does not continue to check the library list of each dependent file to see if there are any duplicates. The complete binding sequence can only be determined at load (`RUN`) time, at which point, the loader may detect a duplicate and fail the load. The loader allows duplicate libraries to be specified if they are adjacent in the expanded binding sequence. In that case, the loader continues the load by *skipping over the adjacent duplicate libraries.* The net effect is that they are reduced to a unique copy of library in the binding sequence.

Note

The link time specified dependency lists in the libraries CANNOT be overridden by the library list specified at run time.

HPGETPROCPLABEL()

The syntax and the semantics of `HPGETPROCPLABEL()` have been enhanced with the introduction of Dependent Libraries.

`HPGETPROCPLABEL` can now be used to obtain the runtime address of a data symbol as well as a code symbol. As before, if the symbol does not exist, the loader tries to load the appropriate executable files to find it. If the loader still cannot find the symbol, it returns with an error.

Note

There are a number of extension parameters being added to HPGETPROCPLABEL. Compatibility with the older versions of this intrinsic is completely maintained for the case where the *firstfile* does not have a dependency list. For the case where the *firstfile* does have a dependency list, the behavior of this intrinsic may not be compatible with earlier versions.

The new calling sequence for HPGETPROCPLABEL is as follows:

Syntax

```
procedure HPGETPROCPLABEL(
   anyvar symbolname     : { CA  };
      var plabel         : { U32 };
   anyvar status         : { I32 };
   anyvar firstfile      : { CA  };
          case_sensitive : { B   };
          symbol_type    : { U32 };   {new parameter}
      var data_size      : { U32 };   {new parameter}
          position       : { U32 };   {new parameter}
          search_path    : { U32 };   {new parameter}
          binding        : { U32 } )  {new parameter}
```

Use

The HPGETPROCPLABEL() intrinsic locates a procedure or a data symbol in an NM executable library file (xl) or the loaded program file and returns its procedure label (for a procedure) or a data label and size (for data). In addition, if the symbol is not yet loaded for the process, the intrinsic dynamically loads the symbol.

The caller can then use the plabel to call the specified procedure dynamically, or if it is a data label, to access the data by a pointer dereference, provided the programming language contains features for this functionality.

Labels returned by HPGETPROCPLABEL() are valid only for the duration of the calling process.

For data symbols from a thread private SOM, the data labels returned are only valid on the particular thread that issued the call.

symbolname

character array (required).

Passes the name of the procedure or the data. This parameter specifies the name of the symbol. The first character of *symbolname* designates the terminating character that HPGETPROCPLABEL() uses to search for the end of the name. The delimiter can appear again only following the last valid character of the symbol name.

plabel

32-bit unsigned integer by reference (required).

Returns a procedure label (NM plabel) for the procedure that was found. If the *symbol_type* refers to a data symbol, the value returned is the address of the data symbol and can be dereferenced to obtain the data's value.

status

32-bit unsigned integer by reference (optional).

The semantics and syntax of this parameter are identical to the existing version. New errors and warnings are documented in the system message catalog `SYSCAT.PUB.SYS`.

firstfile

character array (optional).

Passes the name of the program file or XL at which to begin searching. The first character of *firstfile* designates the terminating character that `HPGETPROCPLABEL()` uses to search the end of the filename. That delimiter can appear again only following the last valid character of the name.

For procedure symbols, the *firstfile* is located in the binding sequences of the calling process. At this point, if the parameter *search_path* specifies that only this file be looked up, then `HPGETPROCPLABEL()` searches only the *firstfile* for the procedure named *symbolname*. If it cannot find the procedure then it returns with an error.

However, if the *search_path* specifies that all subsequent libraries in the binding sequence be searched, then `HPGETPROCPLABEL()` directs the NM Loader to search through each file in the binding sequence to which the *firstfile* belongs for the first instance of *symbolname*. If either *symbolname* is not located or *symbolname* contains unresolved external references that cannot be satisfied in subsequent libraries, an error is returned.

For data symbols, the NM Loader looks for the *symbolname* in the existent loaded environment. It examines all the loaded data symbols to find a match and returns the one it finds. If it cannot find the *symbolname* then it examines the *firstfile* parameter. If the *firstfile* is located in some binding sequence of the calling process, then it returns an error—this means that the *symbolname* doesn't exist.

If *firstfile* is not located in the binding sequences, then the loader starts a new binding sequence independent of all existing binding sequences. Please refer to the "Provisional Binding Sequence" section for more details on how new binding sequences are created.

Default: System Libraries.

Note

The data label returned may be found from some file other than the original *firstfile*. If the *search_path* parameter specifies *search_firstfile_only* and a data label is requested then an error occurs. The loader considers the *search_firstfile_only* parameter an incompatible specification with a data symbol request.

casesensitive

boolean (optional)

The semantics and syntax of this parameter are exactly as before. However, the loader now performs a true non-casesensitive search for symbol names, as documented in the *MPE/iX Intrinsics Reference Manual* (32650-90028).

Important Details Please Read

Prior to MPE/iX Release 5.5, the search algorithm for non-casesensitive symbols would not find mixed-case symbols. Now that the non-casesensitive search performs correctly, previously running applications (including third-party applications), may fail with a Data Memory Protection Trap (DMPT).

For example, an application may fail with a DMPT when calling the `FREAD` intrinsic at `fread+$58` (XL.PUB.SYS). In this case, the application is using the following MPE/iX intrinsic call to search for a non-MPE/iX FREAD in XL.PUB.SYS, if one exists:

`HPGETPROCPLABEL(`*procname, plabel, status, firstfile, casesensitive*`)`

where `procname = *FREAD*`, `firstfile = *XL.PUB.SYS*`, and `casesensitive = FALSE`.

Prior to the casesensitive modification, either a third-party `FREAD` (uppercase) plabel (if one exists in XL.PUB.SYS), or the MPE/iX `FREAD` intrinsic plabel of XL.PUB.SYS was returned to the caller. Since the casesensitive modification, however, the plabel for the ANSI C library `fread()` (lowercase) of XL.PUB.SYS is being returned if a third-party `FREAD` (uppercase) does not exist in XL.PUB.SYS. Later when the application calls the `FREAD` intrinsic, which is bound to the ANSI C `fread()`, the DMPT abort at `fread+$58` results.

In addition to the `FREAD/fread()` naming conflicts, `FOPEN/fopen()`, `FWRITE/fwrite()`, `FCLOSE/fclose()` or any other like naming conventions may experience similar problems.

The solution is to pass `casesensitive = TRUE` since only FREAD" (uppercase) from either XL.PUB.SYS or XL.PUB.SYS is desired and not `fread()` (lowercase) or any mixed-case variation of XL.PUB.SYS.

Default: False

symbol_type

32-bit unsigned integer (optional).

Passes the requested type of symbol.

A value of 0 indicates that the caller is looking for a procedure symbol. In this case, `HPGETPROCPLABEL()` will try to locate a procedure symbol and return its Native Mode (NM) plabel.

A value of 1 indicates that the caller is looking for a data symbol. In this case, `HPGETPROCPLABEL()` will try to locate a data symbol and return its data label (address) and size.

Default: Procedure symbol (0).

data_size

32-bit unsigned integer by reference (optional).

Returns the size of the symbol found.

For procedure symbols, it returns a 0.

For data symbols. It returns the size in bytes of the data symbol whose data label is being returned. (Size is only available for storage request symbols.)

If this parameter is not passed in, then no size information is returned.

Default: Nil

position

32-bit unsigned integer (optional).

Passes the point in the *process binding sequence* were the *firstfile* and its dependency list of libraries should be resolved. There are two conditions which cause this parameter to be ignored. They are the following:

- If the *firstfile* is loaded, or
- If a file in the dependency list of libraries is already loaded in the *process binding sequence*, then the *firstfile* and the unloaded files in its dependency list are loaded through the already loaded file.

Refer to the "Position of Dynamic Libraries" section for more details.

A value of 0 indicates that the *firstfile* and its list of libraries, if any, should be resolved through the system libraries.

A value of 1 indicates that the *firstfile* and its list of libraries, if any, should be resolved through the binding sequence inserted before the program file. This option can be only specified if the program running on the process is an NM program.

Please look at the following sections on binding sequence extensions for more details.

Default: System Libraries (0).

search_path

32-bit unsigned integer (optional).

Passes the search path for the *symbolname*. This parameter is examined only for procedure symbols. A non-zero value for data symbols will return an error.

A value of 0 causes the NM Loader to search the *firstfile* and all subsequent libraries in the binding sequence to which *firstfile* belongs, for the first instance of the procedure specified by *symbolname*.

A value of 1 causes the NM Loader to search only *firstfile* for the procedure specified by *symbolname*. If it is not found in *firstfile* then an error is returned.

Default: Search all libraries in the binding sequence (0).

binding

32-bit unsigned integer (optional).

Passes the type of binding desired. Currently there is only one option in use that is **immediate**. Future options might include **deferred** and **dont_care** options.

Default: Immediate binding (0).

Extensions to the MPE/iX binding sequence

The binding of imported code and data symbols with export symbols is governed by the order in which libraries are loaded by the loader, i.e., a binding sequence. With the addition of Dependent Libraries, MPE/iX can now support more complex binding sequences. Searching accomplishes the task of matching unresolved import functions from the program and succeeding libraries to export functions located in a newly opened library in a left-to-right traversal order. Binding occurs when an export function is located that matches preceding unresolved import function.

Dependency Lists

For program files, the Link Editor supports an option, `XL=` on the `LINK` command. This option allows you to specify an *ordered list of libraries* on which the program is dependent. At run time, when you invoke the program, you do not need to know what library this program needs in order to run. The program has that information within itself. The list of libraries specified in the `XL=` command is the dependency list of libraries for the program file.

This concept has been extended to executable libraries. The Link Editor can now support a new parameter `lib=` on the `BUILDXL` and `ALTXL` commands which allow you to specify a dependency list which is an ordered list of libraries on which the library depends. This way the calling program doesn't need to consider the libraries on which its own Dependent Libraries depend. Due to the proliferation of the software toolkits on MPE (such as DCE, Posix, Encina, or CICS), it was difficult for the application developer to build their software libraries in the forward binding manner.

Another reason for Dependent Libraries is dynamic loading. Today, when a library is dynamically loaded, it can only resolve its symbols through the system libraries (Sys Libs) - XL.PUB.SYS and NL.PUB.SYS. This is very restrictive. The provision of Dependent Libraries allows dynamically loaded libraries to pull in as much code as they like, without the user of the library being aware of it.

Dependency Tree

There are two key questions that are fundamental to Dependent Library developers:

- When a Dependent Library is loaded, in what order are the libraries on which it is dependent, loaded in relation to other Dependent Libraries?
- When Dependent Libraries are added dynamically via a `HPGETPROCPLABEL()`, where are these dynamically loaded Dependent Libraries placed in relation to other already loaded libraries?

In order to explain the process of loading Dependent Libraries, a new structure is defined, a dependency tree, and a new algorithm is implemented, preorder traversal of a dependency tree, which converts a dependency tree into the desired binding sequence.

The dependency tree is formed in the following way:

- Each library with a dependency list is a parent node.
- All the libraries in the dependency list of a library are children of the parent node.
- Libraries without dependency lists are leaf nodes.

Having built the dependency tree from the dependency lists, the loader follows a standard preorder traversal of the dependency tree to generate a *provisional binding sequence* where the list of Dependent Libraries of each file is recursively expanded and checked for internal consistency such as no duplicates allowed. This sequence is then used to determine where to load the libraries (where to attach it to the loader Process File List PFL).

For example, the program Pgm has a dependency list of lib1, lib2, lib3. Lib2 has a dependency list of lib21, lib22. Lib22 has a dependency list lib221, lib3. And, lib1, lib21, lib221 and lib3 have no dependency lists, then the dependency tree would appear as follows:

```
                         Pgm
                          |
    -------------------------------------------
    |                     |                    |
   lib1                  lib2                 lib3
                          |
                 -----------------
                 |               |
                lib21           lib22
                                 |
                            -----------
                            |         |
                          lib221     lib3
```

A *preorder traversal* on this tree would yield the *provisional binding sequence*

```
[ Pgm ->lib1->lib2->lib21->lib22->lib221->lib3->lib3 ].
```

Once the duplicate is removed, the static binding sequence is deduced to the following:

```
Pgm ->lib1->lib2->lib21->lib22->lib221->lib3-> Sys Libs
```

Binding Tree

The following discussion addresses the first question:

> When a Dependent Library is loaded, in what order are the libraries on which it is dependent, loaded in relation to other Dependent Libraries?

Each process has a binding tree associated with it which is composed of a static binding sequence and a number of dynamic binding sequences all attached together.

When a process starts up, a static binding sequence is constructed as explained in the above section. This sequence is now interpreted as a binding tree (quite different from the dependency tree described in the above section).

The binding tree has NL.PUB.SYS as the root. NL.PUB.SYS has one child, which is XL.PUB.SYS. The program file is a leaf node. Its parent is the first library in the static binding sequence. Each library in the sequence is a child of the succeeding library. The last library in the binding sequence has XL.PUB.SYS as its parent. The binding tree may look like this:

```
                  nl.pub.sys
                     /|\
                      |
                  xl.pub.sys
                    ~/|
                     /
                  lib3
                 ~/|
                 /
               lib221
              ~/|
              /
            lib22
           ~/|
           /
         lib21
        ~/|
        /
      lib2
     ~/|
     /
   lib1
  ~/|
  /
Pgm
```

Note

This is different from the dependency tree used in the construction of the binding sequence. There, all the dependencies were direct children of the program file. The binding tree in this section actually inverts the dependency tree and flattens it out.

The dependency tree can be converted into a *process binding sequence* by placing it horizontally and drawing it with forward link arrows. This structure depicts the ordered relationship of libraries with respect to their loading sequence.

```
Pgm->lib1->lib2->lib21->lib22->lib221->lib3-|
      ^     ^      ^      ^      ^      ^   |
      |     |      |      |      |      |   |
     ...   ...    ...    ...    ...    ...  |->Sys Libs
                                            |
                                            |
                                       ... -|
```

The dots with an arrow indicates that a dynamic load can attach to any existing point in the current binding sequence or it can start a separate binding sequence.

Provisional Binding Sequence

The following discussion addresses the next question:

> When Dependent Libraries are added dynamically via a HPGETPROCPLABEL(), where are these dynamically loaded Dependent Libraries placed in relation to other already loaded libraries?

This section explains how Dependent Libraries are dynamically loaded.

If the library has been linked with a list of libraries on which it is dependent, then when it is dynamically loaded, it expects its symbols to be resolved through the list of libraries it had specified at link time.

A *provisional binding sequence* is created by the loader when confronted by a call to HPGETPROCPLABEL() which has a parameter *firstfile*, which turns out to be dependent on a library list dlib1, dlib2, ... dlibn. The *provisional binding sequence* becomes

```
[ firstfile -> dlib1 -> dlib2 -> ... -> dlibn ].
```

Note The Dependent Libraries list is recursively flattened out as described in the section above.

The NM Loader enforces the following rules at dynamic load time.

1. All libraries in the *provisional binding sequence* that are already loaded must be in the same loaded binding sequence. If this condition is not met—the load fails.
2. All loaded libraries must be in the same order on the *provisional binding sequence* and on the existing binding sequence. If this condition is not met, the load fails. The loaded libraries must be in the same order, but there can be other libraries intervening between these libraries.
3. Already loaded libraries and not loaded libraries in the *provisional binding sequence* cannot be interleaved. Going from left to right, every library after the first loaded library must also be loaded— otherwise the load fails.
4. All the security rules remain the same. Namely, a privileged library can only be loaded through privileged libraries.

Rules for Dynamic Libraries - I

Putting it pictorially, let the following be the existing *process binding sequence* at the time of a dynamic load. Note that HPGETPROCPLABEL intrinsic was called previously *n* times to load dlib1 through dlibn.

```
Pgm -> slib1 -> slib2 -> ... slibn -|
                                    |
                             dlib1 -|
                                    |--> Sys Libs
                             dlibn -|
```

Process Binding Sequence at Dynamic Load

Case 1. Consider the case where the user attempts to do a dynamic load of dependent library, lib1, which depends on the list of libraries slib1, slib2, and dlib1. The *provisional binding sequence* is

```
[ lib1 -> slib1 -> slib2 -> dlib1 ]
```

The loaded libraries are slib1, slib2 and dlib1, which are on two different binding sequences. Therefore, Rule 1 for Dynamic Libraries - I is violated—the load will fail.

Case 2. Consider the case where the user attempts to do a dynamic load of lib1 which is dependent on the list of libraries slib2, slib1. The *provisional binding sequence* is

```
[ lib1 -> slib2 -> slib1 ]
```

The loaded libraries are slib2 and slib1. These libraries are already loaded in the order slib1, slib2. Therefore, Rule 2 for Dynamic Libraries - I is violated—the load fails.

Case 3. Consider the case where the user attempts to do a dynamic load of lib1 which is dependent on slib1, newlib, slib2. The *provisional binding sequence* is

```
[ lib1 -> slib1 -> newlib -> slib2 ]
```

The loaded libraries are slib1 and slib2 and satisfy Rule 2 above. However, the intervening library newlib is not loaded. Therefore, Rule 3 for Dynamic Libraries - I is violated—the load fails.

Case 4. Consider the case where the user attempts to do a dynamic load of lib1 which is dependent on slib1, slib3. The *provisional binding sequence* is

```
[ lib1 -> slib1 -> slib3 ]
```

The loaded libraries slib1 and slib3 are on the same binding sequence and occur in the same order. The load will succeed even though there are other libraries intervening between slib1 and slib3, because this is explicitly permitted by Rule 2. The *process binding sequence* would look like the following:

```
Pgm -> slib1-> slib2-> slib3-> ... slibn -|
          ^                               |
          |                               |
        lib1                       dlib1 -|
                                          |--> Sys Libs
                                          |
                                          |
                                   dlibn -|
```

Process Binding Sequence for Case 4

Position of Dynamic Libraries

Up until now, the loader supported only one kind of dynamic binding sequence. All dynamic libraries were loaded through XL.PUB.SYS—that has changed.

The previous section explained the conditions under which a dynamic load will fail. This section explains what happens if the load succeeds. Basically, the loader supports the creation of binding sequences that can attach onto any existing binding sequence.

The loader implements the following rules which dictate where the *provisional binding sequence* is attached to the existing *process binding sequence*.

1. If none of the libraries in the *provisional binding sequence* are loaded, then attach to the system libraries. This creates the new binding sequence:

   ```
   firstfile -> (flattened list of dep.libs) -> Sys Libs
   ```

2. If some of the libraries in the *provisional binding sequence* are already loaded, then attach to the first loaded library in the *provisional binding sequence*.

Rules for Dynamic Libraries - II

Note

In dynamic loading, the starting file is the *firstfile* parameter in `HPGETPROCPLABEL()` intrinsic.

The following cases refer to the existing binding sequences depicted in the figure shown earlier called "Process Binding Sequence at Dynamic Load."

Case 5. Consider the case where the user attempts to do a dynamic load of lib1 which has a dependency list of lib2, dlib1. The *provisional binding sequence* is

```
[ lib1 -> lib2 -> dlib1 ]
```

Since dlib1 is already loaded, the loader creates the new subsequence

```
[ lib1 -> lib2 ]
```

The *process binding sequence* would look like the following:

```
Pgm -> slib1 -> slib2 -> ... slibn -|
                                    |
                                    |
          lib1 -> lib2 -> dlib1 -|
                                    |--> Sys Libs
                                    |
                                    |
                             dlibn -|
```

Process Binding Sequence for Case 5

Case 6. Subsequent to case 5, suppose there was a load of lib11 with a dependency list of lib2 and XL.PUB.SYS. The loader determines that lib2 and XL.PUB.SYS are already loaded on the same binding sequence. It creates a new subsequence [lib11] and loads it through the sequence [lib2, XL.PUB.SYS, NL.PUB.SYS] using Rule 2 for Dynamic Libraries - II.

The *process binding sequence* would look like the following:

```
Pgm -> slib1 -> slib2 -> ... slibn -|
                                    |
                                    |
          lib1 -> lib2 -> dlib1 -|
                   ^                |--> Sys Libs
                   |                |
          lib11 ---|                |
                                    |
                                    |
                             dlibn -|
```

Process Binding Sequence for Case 6

Case 7. Finally, if the user wants to load libxx with a Dependent Library libyy then the loader would create a new sequence [libxx, libyy] and load it though the existing sequence [XL.PUB.SYS, NL.PUB.SYS] using Rule 1 for Dynamic Libraries - II.

The *process binding sequence* would look like the following:

```
Pgm -> slib1 -> slib2 -> ... slibn -|
                                    |
                                    |
             lib1 -> lib2 -> dlib1 -|
                      ^             |--> Sys Libs
                      |             |
             lib11 ---|             |
                                    |
                                    |
                    libxx -> libyy -|
                                    |
                                    |
                             dlibn -|
```

Process Binding Sequence for Case 7

Positional Dynamic Loading

All the discussion on the loading of libraries, etc., has focussed on link time options. There has been no mention of extra control at run time.

This section seeks to give the shared library developers some control over where the dynamically loaded library should load. This is facilitated through an extra parameter that can be passed into `HPGETPROCPLABEL()`.

The extra parameter, *position*, indicates where the *provisional binding sequence* should be attached to the existing process binding tree. It can have an integer value of either **zero** for the system libraries or **one** for the program. If *position* signals the program, then the *firstfile* is loaded through it; otherwise, by default, the *firstfile* is loaded through the system libraries.

Note

There exist two conditions which cause this parameter to be ignored. Please refer to the description of the *position* parameter in the "HPGETPROCPLABEL()" section.

If Dependent Libraries are present, then the name in the *position* parameter is appended to the *provisional binding sequence* created above. This new *provisional binding sequence* is then treated exactly as if it was created using the rules in the above section.

For example, let lib1 have a dependency list of lib2 and lib3. lib2 has a dependency list of lib21.

If the invoker of the dynamic load specifies to load through the program file. then the following *provisional binding sequence* is created:

```
[ lib1 -> lib2 -> lib21 -> lib3 -> Pgm -> ... ]
```

If the invoker of the dynamic load specifies to load through the system libraries. then the following *provisional binding sequence* is created:

```
[ lib1 -> lib2 -> lib21 -> lib3 -> XL.PUB.SYS ]
```

All of the above rules for determining if the load should succeed or not and. if it should succeed. where the new binding sequence should be positioned are applied to all *provisional binding sequence*. The *position* parameter enables the shared library developers to modify link time dependency lists at load time.

Variance of MPE/iX Dependent Libraries and HP-UX Shared Library Dependencies

The new capabilities of MPE/iX have been added in such a manner as to make porting products from HP-UX to MPE/iX as simple as possible. Obviously, these two operating systems are quite different in nature—therefore, some differences exist and are visible to those porting products between the two systems.

The following example shows that care must be taken when porting libraries which have dependency list.

In HP-UX the order in which Dependent Libraries are built is important because a library must exist before you can specify it as a dependency library. This not the case for MPE/iX Dependent Libraries. Consider the following Dependent Libraries built within the HP-UX environment:

```
libQ ->    libB

libD ->    libQ, libB

libP ->    libA, libD, libQ
```

Let's suppose `libP` is loaded via the MPE/iX `RUN` command. The load would terminate because of the rule regarding duplicates.

The reason for the load termination is the following. When loading `libP` in HP-UX, the library search list, as determined by the HP-UX expansion rules, would be the following:

```
libP -> libA -> libD -> libQ -> libB
```

But. for MPE/iX the preorder traversal of the dependency tree would yield the following *provisional binding sequence*:

```
libP -> libA -> libD -> libQ -> libB ->  libB -> libQ
                                  |         |
                                 -adjacent-
```

Pictorially. the MPE/iX dependency tree would be as follows:

```
                    libP
                     |
   -----------------------------------------
   |                 |                    |
  libA              libD                 libQ
                     |                    |
             -----------------           libB
             |               |
            libQ            libB
             |
            libB
```

The duplicate `libB` would be removed because they are adjacent. Hence. the *provisional binding sequence* would be reduced to

```
libP -> libA -> libD -> libQ -> libB ->  libQ
                         |                 |
                        -- not adjacent --
```

But the duplicate `libQ` are not adjacent. therefore. the load would terminate. Actually. including `libQ` as part of `libP`'s dependency list is an unnecessary duplication since it already appears in the `libD`'s dependency list. The resulting porting problem could be resolved by using the Link Editor `ALTXL` command `lib= "libA, libD"`. The following dependency list reduction would simplify the load graph enabling MPE/iX to behave similarly to HP-UX.

```
libP -> libA -> libD -> libQ -> libB
```

Pictorially, the MPE/iX dependency tree would be *reduced* as follows:

```
                    libP
                     |
 --------------------|
 |                   |
libA                libD
                     |
             -----------------
             |               |
            libQ            libB
             |
            libB
```

HP Link Editor/iX Enhancements Detail

by Sue Meloy
Support Technology Center

This article provides more detail on the HP Link Editor/iX enhancements introduced in the article, "HP Link Editor/iX Enhancements Overview," in Chapter 6, "Application Development."

New and Enhanced Link Editor Commands

The following new and enhanced commands were added to support Dependent Libraries:

- The new command, `ALTXL`
- The enhanced command, `BUILDXL`

ALTXL Command

`ALTXL` is a new command in the HP Link Editor/iX that changes the Dependent Library string for an executable library (XL). It currently accepts two options, both required:

- Target XL name
- Dependent library string

Dependent Libraries can be specified in an indirect file or by specifying them directly in the argument to the `LIB=` parameter.

When the Link Editor builds an XL, it inserts an MPE/iX Program Auxiliary Header after the LST header. Any Dependent Library string for the XL is inserted in this header.

The Dependent Library string algorithm for `ALTXL` is similar to the `ALTPROG` string algorithm. If a new Dependent Library string is the same size or smaller than the current Dependent Library string in the XL, the Link Editor overwrites the old string with the new one.

If the new string is larger than the old string, the Link Editor attempts to write the new string at the end of the LST string table. If there is no more room in the string table, an error message is displayed. If this happens, the only way to add/change Dependent Library strings in the XL is to rebuild it.

Syntax:

`ALTXL XL=`*xl_name*`; LIB=`*dependent_library*

Parameters

xl_name	Names the executable library whose Dependent Library list is to be altered.
dependent_library	Names a list of Dependent Libraries that must be loaded when this XL is loaded. Each

Dependent Library must have a filecode of NMXL.

When you want to include several libraries, you can name each library individually, or you can use a single indirect filename. This file should contain a list of the libraries you want to include. Precede the indirect filename with a caret symbol (^).

Note that you must supply at least one Dependent Library since the LIB= parameter is required.

Examples

```
:PRINT MYINDF1
        NEWLIB.LIB.MYACCT
        LIB2.LIB.SYS

:LINKEDIT
LinkEd> ALTXL MYLIB1; LIB=^MYINDF1
LinkEd> ALTXL NEWLIB.LIB.MYACCT; LIB=NEW2.TMP.SYS
LinkEd> EXIT

:RUN MYOUT;XL="MYLIB1"
```

The loader loads the program file, MYOUT. It then loads the executable library, MYLIB1. Since MYLIB1 has Dependent Libraries, the loader loads them: NEWLIB.LIB.MYACCT and LIB2.LIB.SYS.

NEWLIB.LIB.MYACCT has a Dependent Library NEW2.TMP.SYS, so it is loaded after NEWLIB.LIB.MYACCT and before LIB2.LIB.SYS. The load graph is as follows:

```
MYOUT->MYLIB1->NEWLIB.LIB.MYACCT->NEW2.TMP.SYS->LIB2.LIB.SYS
```

```
:LINKEDIT
LinkEd> ALTXL MYLIB2;LIB=MYXL.PUB.LINKER, MYXL2.TMP.SYS
LinkEd> EXIT

:RUN MYOUT;XL="MYLIB2"
```

When MYLIB2 is loaded, the loader loads MYXL and MYXL2. If either of these libraries have their own Dependent Libraries, each new Dependent Library is loaded in order after the library containing the dependency.

BUILDXL Command

BUILDXL now accepts a third parameter, LIB=. This parameter is used to specify Dependent Libraries. This parameter accepts an indirect file or a list of fully qualified library names. The Link Editor concatenates the Dependent Library names into a string and inserts the string into the LST Program Auxiliary header for the loader to search at run time.

Syntax

BUILDXL XL=*xl_file*

[;LIMIT=*max_modules*]

[;LIB=*dependent_library*]

dependent_library	Names a Dependent Library or list of Dependent Libraries that must be loaded when this XL is loaded. Each Dependent Library must have a filecode of NMXL. When you want to include several libraries, you can name each library individually, or you can use a single indirect filename. This file should contain a list of the libraries you want to include. Precede the indirect filename with a caret symbol (^).

If the LIB= option is not specified, the XL is built without any Dependent Libraries.

Example:

```
BUILDXL MYXL2;LIB=^MYINDF
```

BUILDXL creates an executable library called MYXL2 that has Dependent Libraries as specified in MYINDF. When MYXL2 is loaded, its Dependent Libraries are also loaded.

POSIX Libraries in XLs

by Bill Bennett
Commercial Systems Division

This is an update of the "POSIX Libraries in XLs" article for the MPE/iX General Release 5.0 (C.50.00) *Communicator*. This article corrects an error in the examples and adds a new section, "POSIX Shell Scripts," which illustrates how to link using shell scripts.

Intended Audience

With the introduction of *Shared Globals* it is possible to have POSIX C programs use POSIX C executable library files (XLs). It is also possible for POSIX and non-POSIX C programs to have externals restored by both POSIX and non-POSIX XLs.

This article is intended for application developers, porters, support and training personnel. The article provides an overview of how to create and use XLs containing POSIX modules.

Introduction

POSIX C requires global data to be accessible from the user program and library functions. Before Shared Globals (sharing data between programs and XLs) POSIX C applications could not be put into XLs; all POSIX objects and libraries had to be linked into the program. Now with Shared Globals it is possible to put POSIX libraries into XLs.

POSIX in XLs

Manipulating XLs can only be done with the Link Editor. There are three basic methods for putting POSIX modules into XLs. These methods are detailed below. Also discussed is the mixing of POSIX and non-POSIX modules and the caution that must be used when mixing modules.

POSIX and non-POSIX C use the same library calls, for example *open()*. The POSIX *open()* and the non-POSIX C *open()* are different; so there are two *open()*s. XL.PUB.SYS contains the non-POSIX C library and the POSIX RL (/lib/libc.a) contains the POSIX C library.

It is possible to have both POSIX and non-POSIX programs link with both POSIX and non-POSIX XLs for shared services. But care must be taken to make sure neither the program nor the XLs bind to the "wrong" code. Wrong means that a POSIX function was inadvertently called by some non-POSIX code or vice versa.

Method 1

The most bullet proof method of adding POSIX modules to XLs is to link the POSIX RL into each POSIX program and merge the POSIX RL into every load module in an XL. The POSIX RL is released with its entry points hidden so the POSIX code cannot be inadvertently called by non-POSIX programs and load modules. But, to make sure the entry points are hidden it is good practice to perform a HIDERL on the POSIX RL when creating your XLs.

With all POSIX library entry points hidden. non-POSIX programs will bind to the entry points of the non-POSIX libraries. This allows old non-POSIX programs to work the same and allows POSIX programs to access non-POSIX utilities in XLs (like database access). POSIX and non-POSIX programs can access POSIX utilities in XLs. Also, since the POSIX programs and XLs resolve all their own dependencies, they cannot be harmed by any changes in the library link order.

With this method it is also possible to have POSIX and non-POSIX modules in a single XL. The POSIX modules simply have to merge in the POSIX RL and the non-POSIX modules do not. For example,

```
:COPY /lib/libc.a, LIBC

:COPY /lib/libm.a, LIBM

:COPY /lib/libsvipc.a, LIBSV

:LINKEDIT

LinkEd> HIDERL RL=LIBC;ALL

LinkEd> HIDERL RL=LIBM;ALL

LinkEd> HIDERL RL=LIBSV;ALL

LinkEd> PURGERL ENTRY=_start;RL=LIBC

LinkEd> LINK FROM=progo;to=prog;RL=LIBC,LIBM;XL=appxl;SHARE

LinkEd> BUILDXL XL=appxl;LIMIT=10

LinkEd> ADDXL FROM=appo;RL=LIBC,LIBM,LIBSV;MERGE;SHARE

LinkEd> ADDXL FROM=app2o;RL=LIBC;MERGE;SHARE

LinkEd> EXIT
```

Note

In the previous example, `libm.a` and `libsvipc.a` are used in addition to `libc.a`. They are only required if the program or load module call the functions they contain. In the later examples only `libc.a` will be used but the other libraries could be used if necessary.

The Link Editor does not support HFS filenames so all examples assume MPE filenames.

Method 2

Another method of adding POSIX modules to XLs is to reveal the POSIX library entry points in the application XL and bind the application program to this one POSIX library. This method reduces program size and makes updating the POSIX library simple because it is in only one place. Only POSIX programs can call this XL. Any non-POSIX XL utilities must be added to the XL list after the XL that contains the POSIX routines (`appxl` in the below example.) For example,

```
:COPY /lib/libc.a,LIBC


:LINKEDIT

LinkEd> REVEALRL RL=LIBC;ALL

LinkEd> EXTRACTRL ENTRY=_start;FROM=LIBC;TO=starto

LinkEd> PURGERL ENTRY=_start;RL=LIBC

LinkEd> LINK FROM=progo,starto;to=prog;XL=appxl;SHARE

LinkEd> BUILDXL XL=appxl;LIMIT=10

LinkEd> ADDXL FROM=appo;MERGE;SHARE

LinkEd> ADDXL FROM=appo2;MERGE;SHARE

LinkEd> ADDXL FROM=LIBC;MERGE;SHARE

LinkEd> EXIT
```

Note

`_start` needs to be in every program. (You can link all of `libc.a` into the program but by just including `_start`, the program size is reduced.)

Method 3

This last method is just a variant of the previous method. In this method the POSIX RL is made into its own XL. This implementation is a little more tricky because it requires a specific order in the XL list (if additional XLs are necessary). This method reduces program size and makes updating the POSIX library very easy because it is in a separate file. Only POSIX programs can call these XLs. Any non-POSIX XL utilities can only be added to the XL list after the POSIX XL (`cxl` in the below example.) But problems may occur if a user uses `XL=` on the `RUN` command or if an XL is missing.

For example.

```
:COPY /lib/libc.a,LIBC


:LINKEDIT

LinkEd> REVEALRL RL=LIBC;ALL

LinkEd> EXTRACTRL ENTRY=_start;FROM=LIBC;TO=starto

LinkEd> PURGERL ENTRY=_start;RL=LIBC

LinkEd> LINK FROM=progo,starto;to=prog;XL=appxl,cxl;SHARE

LinkEd> BUILDXL XL=appxl;LIMIT=10

LinkEd> ADDXL FROM=appo;MERGE;SHARE

LinkEd> ADDXL FROM=appo2;MERGE;SHARE

LinkEd> BUILDXL XL=cxl;LIMIT=10

LinkEd> ADDXL FROM=LIBC;MERGE;SHARE

LinkEd> EXIT
```

Note The XL order is important. the POSIX XL must be last.

POSIX Shell Scripts

All the previous examples can be performed using POSIX Shell Scripts. The user can use these scripts from the shell and not have to switch between the MPE/iX CI and the POSIX Shell. Below are three utility scripts that will be used to implement the examples.

Note

These utility scripts **must** be run from an MPE/iX Group.

libc.sh

This script creates three files out of the `/lib/libc.a` file.

LIBC	A version of `/lib/libc.a` without `_start`
STARTO	An object file containing only `_start`
CXL	An XL file containing `/lib/libc.a` without `_start`

These files will be used in the other utility scripts as well as in the further examples.

```
#
# put commands in a temporary file to be read in with LINKEDIT
#
rm -f TMPFL
echo ':purge LIBC' >> TMPFL
echo ':purge starto' >> TMPFL
echo ':purge cxl' >> TMPFL
echo ':copy /lib/libc.a,LIBC' >> TMPFL
echo ' revealrl rl=LIBC;all' >> TMPFL
echo ' extractrl entry=_start;from=LIBC;to=starto' >> TMPFL
echo ' purgerl entry=_start;rl=LIBC' >> TMPFL
echo ' buildxl xl=cxl;limit=10' >> TMPFL
echo ' addxl from=LIBC;merge;share' >> TMPFL
#
# invoke linkedit to build the POSIX RL (/lib/libc.a)
#
callci "linkedit < TMPFL"
#
# purge the temporary file
#
rm -f TMPFL
```

ld.sh

This script builds an XL file and loads it with object files.

```
if test "$#" = 0
then
        echo "Usage:"
        echo "$0 xlfile objfile objfile ..."
        echo "  xlfile is the Shared Library to be created"
        echo "  objfiles are the Object files to be added"
        exit
        fi
#
# This function adds a POSIX prefix if necessary
#
fixup() {
if [ "$1" = "${1#/}" ]
       then
       if [ "$1" = "${1#\.}" ]
               then
                       echo "./$1"
               else
                       echo $1
               fi
       else
               echo $1
fi
}
#
# Setup xlname
#
xlname=`fixup "$1"`
#
# create the XL library file if it doesn't exist
#
callci "file xlname=$xlname"
if [ ! -f "$xlname" ]
then
        callci "linkedit 'buildxl xl=*xlname;limit=10'"
fi
#
# put the object files in a temporary file to be read in
# with LINKEDIT
#
rm -f TMPFL
shift
for i in $*
do
        fixup $i >> TMPFL
done
#
# add the object files to the XL
#
callci "linkedit 'addxl from=^TMPFL; to=*xlname;share;merge'"
#
# purge the temporary file
#
rm -f TMPFL
callci "reset xlname"
```

lk.sh

This script links a program using `STARTO` and not `/lib/libc.a`. This is necessary because c89 always includes `/lib/libc.a`.

```
if test "$#" = 0
then
        echo "Usage:"
        echo "$0 progfile xllist objfile objfile ..."
        echo "  progfile is the program file to be created"
        echo "  xllist is a list of XLs in quotes"
        echo "  objfiles are the Object files used to make the program"
        exit
        fi
#
# This function adds the POSIX prefix if necessary
#
fixup() {
if [ "$1" = "${1#/}" ]
        then
        if [ "$1" = "${1#\.}" ]
                then
                        echo "./$1"
                else
                        echo $1
                fi
        else
                echo $1
fi
}
#
# setup file equation
#
lkname=`fixup "$1"`
rm -f $lkname
callci "file lkname=$lkname"
#
# get the xllist
#
shift
XL="$1"
#
# put the object files in a temporary file to be read in with LINK
#   start by putting STARTO (_start) in first
#
echo STARTO > TMPFL
shift
for i in $*
do
        fixup "$i" >> TMPFL
done
#
# LINK the program
#
callci "link from=^TMPFL; to=*lkname;share;CAP=PH;XL='$XL '"
#
# purge the temporary file
#
rm -f TMPFL
callci "reset lkname"
```

Note

`ld.sh` and `lk.sh` use the POSIX HFS names. The HFS names should only be used with the Link Editor through file equations and indirect files.

The filenames in the XL list of a link must use MPE syntax or, if using HFS syntax, it must be the absolute path (one that starts with /). Relative paths (ones that start with ./) will not work. The user must enter the entire path (as in Method 1 below) or preappend the current path (as in Methods 2 and 3 below).

These scripts are examples of how to use scripts with the Link Editor. There is no error checking and conflicts may occur with **TMPFL**. You are encouraged to modify them for your use.

Before using the following methods, invoke `libc.sh` to create the LIBC file.

Method 1

The following implements Method 1 using POSIX Shell Scripts.

```
callci "LINKEDIT 'HIDERL RL=LIBC;ALL'"

c89 prog.o -o prog -lm "-WL,XL=/TEST/PUB/appxl"

ld.sh appxl appo LIBC /lib/libm.a /lib/libsvipc.a

ld.sh appxl app2o LIBC
```

Note

c89 requires that the object file have an extension of `.o` to work.

Method 2

The following implements Method 2 using POSIX Shell Scripts.

```
lk.sh prog $PWD/appxl progo

ld.sh appxl appo

ld.sh appxl appo2

ld.sh appxl LIBC
```

Method 3

The following implements Method 3 using POSIX Shell Scripts.

```
lk.sh prog "$PWD/appxl,$PWD/cxl" progo

ld.sh appxl appo

ld.sh appxl appo2

ld.sh cxl LIBC
```

Constraints

Programs and libraries that wish to mix POSIX and non-POSIX C must adhere to the following constraints:

1. No file descriptors may be shared across library boundaries. In particular, file descriptors may not be passed as arguments to functions in other libraries. For example a POSIX program cannot open a file and then pass the file descriptor to a non-POSIX XL function, expecting the function to access the opened file. Exceptions to this constraint are files 0, 1, and 2 (**stdin**, **stdout**, and **stderr**)
2. Environment variables may not be used for global data. The *getenv()* function behaves differently in the two libraries.
3. The C **pid** (returned from **getpid**) may not be shared across library boundaries. The libraries return different types.

Related Topics

For a complete description of the new Shared Global Data functionality please read the *HP Link Editor/iX Technical Addendum* (32650-09476).

Please refer to the following C.50.00 *Communicator 3000 MPE/iX General Release 5.0* articles for a better understanding of Shared Globals functionality.

- "Introducing Shared Globals on MPE/iX Loader" in the "Application Development" chapter for an overview of the Shared Globals benefits and features.
- "MPE/iX Shared Globals Overview" in the "Technical Articles" chapter for more detailed information about Shared Globals and its related components.

STORE and TurboSTORE/iX 7x24 True-Online Backup New Functionality

by Jim Nissen
Commercial Systems Division

Overview

This article provides more detailed information about the new STORE and TurboSTORE/iX 7x24 True-Online Backup functionality and enhancements introduced in the article, "Introducing TurboSTORE/iX 7x24 True-Online Backup," in Chapter 3, "System Management." For more in depth information, see the *STORE and TurboSTORE/iX Products Manual* (B5151-90001).

The following functionality has been added to support the TurboSTORE/iX 7x24 True-Online Backup product:

- New TurboSTORE/iX 7x24 True-Online Backup options: `ONLINE` (with `START`, `END`, `ASK` parameters), and `LOGVOLSET`
- New STORE options: `PARTIALDB`. `FULLDB`. `STOREDIRECTORY`, `NOSTOREDIRECTORY`, and `STATISTICS`
- Storing to disk files
- Restoring from disk files

The following enhancements have been added to support the TurboSTORE/iX 7x24 True-Online Backup product:

- Physical consistency is ensured for system files, some flat files, and any other files not associated with a database.
- Shadow log files can now be created on a specified volume set.

The descriptions of the new functionality and enhancements follows.

New TurboSTORE/iX 7x24 True-Online Backup Options

ONLINE

A 7x24 true-online backup is performed only when either the `START` or the `END` parameter is included with the `ONLINE` option. If no parameters are specified for `ONLINE`. the online backup is used, which requires that all files be closed for write access at that time. The online backup only allows for the sync point to be at the beginning of the store.

Up to 15 TurboSTORE/iX 7x24 True-Online backups can be running at once. Also. a file can be stored by up to 15 7x24 true-online backups.

The syntax for ONLINE is as follows:

$$\texttt{;ONLINE}\left[=\left\{\begin{array}{l}\texttt{START}\\ \texttt{END}\end{array}\right\}[\texttt{,}\textit{time}][\texttt{,ASK}]\right]$$

- **START and END**

  The START and END parameters control when the back up occurs by using a *sync point*. To sync means to establish a reference point in time where all files are synchronized. When the files on the backup are restored, they look exactly as they did at the moment of the sync point.

  By default, the sync point is at the beginning, ONLINE=START. ONLINE=END places the sync point at the end of the backup

  In addition, to the START or END parameter, you can specify a time for the sync point to occur. The time is specified in 24-hour format (hh:mm[:ss]). If a time is specified with START, the sync point occurs at the time specified, or once all files are prepared for the backup, which ever event occurs last. If the time is specified with END, the sync point occurs at the time specified or once all files have been stored, which ever event occurs last.

- **ASK**

  TurboSTORE/iX 7x24 True-Online Backup now guarantees both physical and logical consistency of database backups. This is done by setting the sync point to occur after all open transactions are complete. Database access is stopped long enough for this to occur.

  Only TurboIMAGE and ALLBASE/SQL databases are automatically quiesced, which means you may have to manually stop access to some non-HP databases. This is done by using the ASK parameter with the ONLINE option to make TurboSTORE wait at the sync point for you to manually quiesce other types of databases before continuing a backup.

LOGVOLSET

TurboSTORE/iX 7x24 True-Online Backup saves the changes made to databases and files being stored in special log files. These files can become quite large when there is a lot of activity on the system. The LOGVOLSET option allows you to specify which volume set to use for the log files. This option can be used either with the online backup, or the 7x24 true-online backup.

New STORE Options

PARTIALDB

You can now easily store databases with 7x24 true-online backup. By default, STORE does **not** allow you to specify an incomplete TurboIMAGE or ALLBASE/SQL database to be stored. However, you can use the `PARTIALDB` option to backup up a fileset even if it contains a partial database. STORE will backup only those dataset files specified in the fileset list.

Caution

If some of the dataset files are missing from the database, the quiesce may not succeed. STORE will still back up the dataset files listed in the fileset list.

Since quiescing a partial database cannot be guaranteed, we recommend that **`PARTIALDB` not be used** with a 7x24 true-online backup.

Specification of the TurboIMAGE root file or the ALLBASE/SQL DBE file causes STORE to store the entire database. If a dataset file is specified without the root file, then STORE prints a warning such as the following:

```
PARTS22.GROUP.ACCT NOT STORED: FILE IS PART OF AN IMAGE
DATABASE AND ROOT IS NOT SPECIFIED
```

If any of the dataset files are specified in addition to the root file, no warning will be displayed but the individual dataset files will be counted as redundantly specified files.

FULLDB

In a non-7x24 true-online backup environment (i.e., `ONLINE=START` or `ONLINE=END` is not specified), you can use `FULLDB` to backup the entire database by just specifying the root filename only in the fileset. By default, STORE allows a partial database backup of only those database files included in the fileset list.

STOREDIRECTORY and NOSTOREDIRECTORY

TurboSTORE/iX 7x24 True-Online Backup now has the ability to put a copy of a backup's STORE label and STORE directory into a disk file using the `STOREDIRECTORY` option. When creating a 7x24 true-online backup with the sync point at the end of the backup, the `STOREDIRECTORY` option is enabled by default. To override this behavior, specify the `NOSTOREDIRECTORY` option.

Note

It is strongly recommended that you use the `STOREDIRECTORY` option when creating 7x24 true-online backups with the sync point at the end of the backup. Doing so greatly increases recovery time when the backup needs to be restored.

This directory information allows RESTORE to more quickly determine where files are located on the backup, and to prompt users to mount the necessary media. The disk file is created by STORE and named using the following convention:

```
store_yyyymmdd_hhmmsstt_pin##_day
```

The file is located in the HFS directory SYS/HPSTORE/store_dirs/.

Important Details Please Read

It is important to note that the HPSTORE group and store_dirs directory do NOT exist on the system by default. You **must** create them, and set up the appropriate security before using the `STOREDIRECTORY` option. STORE will **not** print a warning if it cannot create the file because the path does not exist.

STATISTICS

An additional feature called `STATISTICS` has been added. This option enables customers to get extra data about the backup, which was previously not available. This extra information includes:

- Amount of data written to each piece of media for each parallel set.
- Amount of time required to write each piece of media.
- Throughput for each piece of media.
- Number of retries incurred for each piece of media.

If an online or 7x24 true-online backup is performed, then information on the amount of log data written is displayed.

Storing To Disk Files

You now have the ability to have the output device for a STORE be a disk file. This gives you the ability to store the data to disk for those unique situations where this is desired. Since the disk backup file created is a flat, binary disk file, it can be moved to different systems and then RESTORE can be used to extract the archived files.

File equations are used to tell TurboSTORE/iX 7x24 True-Online Backup to store to a disk file, such as:

```
FILE BACKUP1;DEV=DISC
```

When specifying the file equation, `;DEV=DISC` **must** be specified to STORE to a disk file. The target of the file equation can point to any MPE or HFS filename. Any other information specified in the file equation is ignored by TurboSTORE/iX 7x24 True-Online Backup.

Important Details Please Read

It is **STRONGLY** recommended that you do **NOT** build the target disk file before invoking STORE. TurboSTORE creates the target disk file as a binary file, with a default record size of 256 bytes. The file is also created with the file code of 2501, which is a file system mnemonic for STORE.

Restoring From Disk Files

You can restore files from:

- A disk file created by TurboSTORE by specifying file equations for the disk files. These file equations follow the same rules that TurboSTORE used when it created the disk files.
- Multiple disk files by creating only one file equation that points to the first disk file. Then RESTORE automatically opens the subsequent disk files.
- Parallel disk files by using the RESTORESET option.

Note

Restoring data from disk files created with TurboSTORE/iX 7x24 True-Online Backup is only supported on systems running MPE/iX Release 5.5 or higher. Although only the TurboSTORE/iX 7x24 True-Online Backup product can create disk files, any version of STORE or TurboSTORE running on MPE/iX 5.5 or later can restore data from disk files.

Physical Consistency

Physical consistency is ensured by providing a snapshot of the file set at a specific point in time, specifically, at the sync point. This is adequate for files that are not "related" to other files, and have no dependencies on changes to other files. This includes system files, some flat files and any other files not associated with a database.

For other file types such as KSAM/XL files, TurboSTORE/iX 7x24 True-Online Backup guarantees that no partial transactions are included in the copy of the file stored on the tape.

For native mode message files, 7x24 true-online backup follows the necessary steps to successfully imitate closing the file to ensure data integrity, physical consistency, and recoverability.

User-mapped files are still managed the same way as before. If a user-mapped file is left open for write access across the sync point, then 7x24 true-online backup captures the last known state of the file, which would either be at the last FOPEN, FCLOSE, or FCONTROL 6. If 7x24 true-online backup does not capture any of these events, the file is stored in whatever state it was in when 7x24 true-online backup opened the file, which may be inconsistent. A warning message is printed for all user-mapped files open across the sync point.

Note

Compatibility mode (CM) files, namely Circular (CIR) files, and Relative IO (RIO) files, have internal buffer structures that cannot be logged by TurboSTORE. If these files are open for write access during the sync point, TurboSTORE captures their last known state.

Shadow Log Files

Another new feature of TurboSTORE/iX 7x24 True-Online Backup is the ability to specify where *shadow log* files reside. Shadow log files contain the *before* images of changes to files that are being backed up.

By default, the before image log files are created on the same volume set as the files being stored. You can create the log files on a specific volume set by specifying the volume set name using the `LOGVOLSET` option.

New Functionality in VPlus

by Kumar KN
Commercial Systems Division - India

Introduction

This article provides more detailed information on the following VPlus (version B.06.06) enhancements introduced in the "VPlus Enhancements" article in Chapter 6, "Application Development:"

- Copying of Processing specifications
- Cursor position sensing

Copying of Processing Specifications

The processing specification copy function allows processing specifications to be copied from one field to another field within and across forms files. Processing specifications are entered from the Field Menu of the FORMSPEC utility. This function is also invoked interactively from the Field Menu.

To use this function, enter one of the following commands in the first line of the processing specifications area in the Field Menu:

- `#COPYTO` *newfilename*
- `#COPYFROM` *oldfilename*

If processing specifications exist on the first line, press (Insert Line) before entering the command. After entering one of these commands, press `F3` to execute the command.

Note

These commands are not casesensitive.

#COPYTO

When `#COPYTO` *newfilename* is executed, the processing specifications defined for the field are copied to the file specified by *newfilename*. As the template suggests, the file specified should be a new file. If there are no processing specifications to copy, the new file is not created. The command should then be deleted from the processing specifications area. If any error is encountered in executing the command, it is displayed in the upper area of the Field Menu screen.

#COPYFROM

When `#COPYFROM` *oldfilename* is executed, the processing specifications (if any) in the file specified by *oldfilename* are copied into the processing specifications area of the field. Press (Enter) after the command executes (successfully) to save the copied processing specifications for the field.

Since this command is used to retrieve the processing specifications, it cannot be used if specifications already exist for the field. If any error is encountered in executing the command, it is displayed in the upper area of the Field Menu screen.

Error Messages

The following error messages may be displayed when executing the `#COPYTO` *newfilename* command:

- If the file specified by *newfilename* already exists, the following message is displayed and no copying is done:

  ```
  File already exists!
  ```

- If the syntax of the command is not proper, the following message is diplayed:

  ```
  Check syntax: #COPYTO newfilename.
  ```

- If the file attributes check for a new file results in an error, the following message is displayed:

  ```
  New file check failed!
  ```

  The specified file should be a non-existent permanent file.

- If the file cannot be opened, the following message is diplayed:

  ```
  Can not create new file!
  ```

The following error messages may be displayed when executing the `#COPYFROM` *oldfilename* command:

- If *oldfilename* cannot be opened, the following message is displayed:

  ```
  Unable to open old file!
  ```

- If the syntax of the command is not proper, the following message is displayed:

  ```
  Check syntax: #COPYFROM oldfilename.
  ```

- If processing specifications exist for the field, the following message is displayed:

  ```
  Cannot do a #COPYFROM if proc specs already exist.
  ```

When a new file is created with the `#COPYTO` command, it is created in the permanent file domain. Using this file in a subsequent `#COPYFROM` command does not purge this file. Purge this file manually if it is no longer required.

Cursor Position Sensing

The cursor position sensing function includes three new intrinsics that have been designed to enable application programs to sense the cursor position on a VPlus screen:

`VARMSCP`	Arms or disarms cursor sensing capability.
`VGETSCPFIELD`	Returns information about the cursor position by field number on a VPlus screen.
`VGETSCPDATA`	Returns information about the cursor location by field number and row and column number on a VPlus screen.

These intrinsics enable application programs to determine where the cursor was left on a VPlus screen after a read operation. The information returned to the application varies with the intrinsic used.

Usage

These intrinsics should be called in an application at certain points of the transaction flow:

- Call `VARMSCP` at the *beginning* of **each** VPlus transaction where cursor information is required. Typically, this is prior to a call to `VSHOWFORM` and preceding `VREADFIELDS`.
- Call `VGETSCPFIELD` or `VGETSCPDATA` at the *end* of a VPlus transaction to collect the cursor position or location information. This is *after* a call to `VREADFIELDS`.

Important Details Please Read

- Cursor position sensing is available on DTC-connected terminals only.
- These intrinsics work only on terminals that have the capability to return cursor location information. These terminals are:

700/9X	2392A
	2394A

Intrinsic VARMSCP

Arms or disarms cursor sensing capability.

Syntax

```
VARMSCP {comarea,scpenable}
```

Parameters

comarea Must be *comarea* name specified when the forms file was opened with `VOPENFORMF`. If not already set, the following *comarea* items must be set before calling `VARMSCP`:

cstatus	Set to zero.

comarealen	Set to total number of two-byte words in *comarea*. Must be at least 70 words in length.

VARMSCP may set the following *comarea* item, *cstatus*.

cstatus	Set to nonzero value if call is unsuccessful.

scpenable Two-byte logical variable which determines whether the cursor position sensing is enabled or not.

0 - Disarm cursor sensing.

1 - Arm cursor sensing.

Discussion

When cursor sensing is armed, the data returned after a read, as in VREADFIELDS, is prefixed with an escape sequence which contains the position of the cursor on the screen when the read terminated. This information is used by VGETSCPFIELD and VGETSCPDATA to retrieve the cursor position.

When cursor sensing is disarmed, no cursor position information is available following a read. VREADFIELDS automatically disarms cursor sensing.

VARMSCP should be called *prior* to a cursor sensing transaction to arm the cursor sensing function for the next read. This deletes any existing cursor location information.

Example

```
COBOL
CALL "VARMSCP" USING COMAREA SCP-ENABLE.

SPL
VARMSCP(COMAREA,SCP'ENABLE);
```

These calls arm/disarm cursor sensing depending on whether the second parameter contains a 1 or a 0 respectively.

Intrinsic VGETSCPFIELD

Returns information about the cursor position by field number on a VPlus screen.

Syntax

```
VGETSCPFIELD { comarea,fieldnum }
```

Parameters

comarea	Must be *comarea* name specified when the forms file was opened with **VOPENFORMF**. If not already set, the following *comarea* items must be set before calling **VGETSCPFIELD**:	
	cstatus	Set to zero.
	comarealen	Set to total number of two-byte words in *comarea*. Must be at least 70 words in length.
fieldnum	Two-byte integer variable to which **VGETSCPFIELD** returns the field number of the field where the cursor was last positioned when the read terminated.	

Discussion

When **VREADFIELDS** terminates, the cursor position on the screen is tracked and retrieved by **VGETSCPFIELD**. The information contains the field number of the field in which the cursor was positioned when the read was terminated. No cursor position information is available if a **VREADFIELDS** retry occurs.

VGETSCPFIELD returns a -1 in *fieldnum* if:

- The cursor was not positioned within a field when the read terminated.
- No cursor position information is available when **VGETSCPFIELD** is called.
- The cursor is positioned within a multi-line field **and** beyond the first line of the field.

VGETSCPFIELD should be called *after* each **VREADFIELDS** since it first retrieves the cursor position information, then deletes the cursor position information upon procedure completion.

Example

```
COBOL
CALL "VGETSCPFIELD" USING COMAREA FIELDNUM.

SPL
VGETSCPFIELD(COMAREA,FIELDNUM);
```

These examples return the cursor position on the last read in the *fieldnum* variable.

Intrinsic VGETSCPDATA

Returns information about the cursor location by field number and row and column number on a VPlus screen.

Syntax

```
VGETSCPDATA {comarea,fieldnum,screenrow,screencol}
```

Parameters

comarea	Must be *comarea* name specified when the forms file was opened with VOPENFORMF. If not already set, the following *comarea* items must be set before calling VGETSCPDATA:
	cstatus — Set to zero.
	comarealen — Set to total number of two-byte words in *comarea*. Must be at least 70 words in length.
fieldnum	Two-byte integer variable to which VGETSCPDATA returns the field number of the field where the cursor was last positioned when the read terminated.
screenrow	Two-byte integer variable to which VGETSCPDATA returns the physical row number on the screen where the cursor was last positioned when the read terminated.
screencol	Two-byte integer variable to which VGETSCPDATA returns the physical column number on the screen where the cursor was last positioned when the read terminated.

Discussion

When VREADFIELDS terminates, the cursor location on the screen is tracked and retrieved by VGETSCPDATA. The information contains the field and row and column in which the cursor was positioned when the read was terminated. No cursor position information is available if VREADFIELDS retry occurs.

VGETSCPDATA returns the physical position of the cursor on the screen by row and column number. Rows are numbered from top to bottom and columns are numbered from left to right.

Note The row and column information returned by VGETSCPDATA is *raw* physical information that may not directly correspond to actual elements of the user interface.

Like VGETSCPFIELD, VGETSCPDATA returns a -1 in *fieldnum* if:

- The cursor was not positioned within a field when the read terminated.
- No cursor position information is available when VGETSCPDATA is called.
- The cursor is positioned within a multi-line field **and** beyond the first line of the field.

VGETSCPDATA should be called *after* each VREADFIELDS since it first retrieves the cursor position information, then deletes the cursor position information upon procedure completion.

Example

```
COBOL
CALL "VGETSCPDATA" USING COMAREA
FIELDNUM SCREENROW SCREENCOL.

SPL
VGETSCPDATA(COMAREA,FIELDNUM,
SCREENROW,SCREENCOL);
```

These examples return the cursor position on the last read in the *fieldnum*, *screenrow*, and *screencol* variables.

AIF Enhancement to AIFPROCGET Details

by John Berry
Commercial Systems Division

Overview

This article provides more detailed information on the AIFPROCGET enhancement, introduced in the article, "AIF Enhancement to AIFPROCGET Overview," in Chapter 3, "System Management."

AIFPROCGET, item *2149*, was enhanced to retrieve information on sockets owned by a given process (pin or pid). This item can be used to request information on all or just incoming socket connections belonging to a process. The maximum number of sockets that can be opened per process is currently 1,024.

connection_type Record Type

A pointer to the record type, *connection_type*, must be passed into the AIFPROCGET parameter list. The caller is responsible for initializing the *connection_type*. Each entry has the following type definitions:

```
connection_type =

   packed record
     cnt     : integer;
     mode    : integer;
     entry   : array[1..n] of conn_element_type;
   end;

Record size:  4 + 4 + 20n bytes
Alignment  :  4 bytes

n represents a user-defined size.
```

Field descriptions for the *connection_type* record structure are as follows:

cnt	Maximum number of socket connections to return. On input, the *cnt* field represents the number of allowable elements in the entry array. On return, the *cnt* field will contain the actual number of socket connections.
mode	Determines which type of socket connections to return. Currently, it returns all or just incoming connections. The *mode* field can be initialized with one of two constants:

0	When *0* is passed, all connections are returned.
1	When *1* is passed, incoming connections are returned.

conn_element_type	Connection information record. See the next section, "conn_element_type Field," for more details.

Once AIFPROCGET returns, an array of connection entries are available in the *connection_type* record structure. Each entry has the following type definition:

```
conn_element_type =

                    packed record
                      local_ip_addr    : bit32;
                      remote_ip_addr   : bit32;
                      local_sap        : bit16;
                      remote_sap       : bit16;
                      socket_state     : shortint;
                      flags            : flag_type;
                      protocol         : bit8;
                      socket_type      : bit8;
                      filler           : bit16;
                    end;
```

conn_element_type Field

Field descriptions for the *conn_element_type* record structure are as follows:

local_ip_addr	IP address of the local HP 3000.
remote_ip_addr	IP address of the remote machine.
local_sap	local TCP port number in use.
remote_sap	TCP port number allocated on remote machine.
socket_state	Socket state.
flags	*flag_type* record.
protocol	Protocol being used.
socket_type	Type of socket being used.

The *flags* field in *conn_element_type* has the following record definition:

```
flag_type =

            packed record
              case integer of
                1: (all                   : bit16);
                2: (select_supported      : boolean;
                    bsd_sk                : boolean;
                    catch_all_flag        : boolean;
                    tcp_msg_mode          : boolean;
                    connection_initiator  : boolean
                    filler                : bit11);
            end;
```

flags Field

Field descriptions for the *flag_type* record structure are as follows:

select_supported	True, if the *select()* function is supported on this socket.
bsd_sk	True, if the connection is through a BSD Socket. False, if the connection is through a NetIPC Socket.
catch_all_flag	True, if this is a NetIPC Socket using X.25 level 3 direct access and the socket was created with the *catch_all_flag* set.
tcp_msg_mode	True, if the connection is using TCP in message mode.
connection_initiator	True, if the local process initiated the connection.

The *protocol*, *socket_type* and *socket_state* fields in *conn_element_type* will return one of the following values:

```
PROTOCOLS

  TRANSMISSION CONTROL PROTOCOL = 4
  USER DATAGRAM PROTOCOL        = 5
  PACKET EXCHANGE PROTOCOL      = 6

TYPES OF SOCKETS

  CONNECTION SOCKET = 0
  DATAGRAM SOCKET   = 1
  CALL SOCKET       = 3

SOCKET STATES

  INITIALIZED  = -1
  CONNECTING   =  1
  WAIT_CONFIRM =  2
  DUPLEX_OPEN  =  3
  SIMPLEX_IN   =  4
  SIMPLEX_OUT  =  5
  CLOSED       =  6
  CONN_ABORTED =  7
  BOUND        =  8
  LISTENING    =  9
  SHUT         = 10
  LINGERING    = 11
```

socket_state Field

Descriptions for the values of the *socket_state* field are as follows:

INITIALIZED	Connection doesn't exist.
CONNECTING	IPCCONNECT called, waiting for IPCRECV.
WAIT_CONFIRM	IPCRECVCN done, user must accept/reject.
DUPLEX_OPEN	Connected, can send and receive data.
SIMPLEX_IN	Graceful close. can only receive.
SIMPLEX_OUT	Received graceful close, send only.
CLOSED	Both parties closed. user must shut down.
CONN_ABORTED	Connection aborted, user must shut down.
BOUND	Socket is bound to an address.
LISTENING	*sk_listen* has been called.
SHUT	Socket shutdown or encountered error.
LINGERING	Socket shutdown lingering pending.

HP Patch/iX Technical Overview

by Deb Alston
Commercial Systems Division

Introduction

This article provides a technical overview of the new MPE/iX patch management tool, HP Patch/iX introduced in the article. "Introducing HP Patch/iX: A New MPE/iX Patch Management Tool," in Chapter 3, "System Management." HP Patch/iX provides a user-friendly environment for customization and installation of Hewlett-Packard supported patches on an MPE system.

Technical Overview

In the new MPE/iX patching environment, patching can be viewed as two distinct phases:

Phase I

Phase 1 does the following:

- Selects and qualifies the patches.
- Creates the CSLT/STORE tape (or STORE only tape, assuming no patches are being applied to the MPE operating system).
- Uses a screen-based interface with viewers that provide extensive patch information about each of the patches in the set, and can be done with users on the system.
- Provides the flexibility of installing a single reactive patch, multiple reactive patches from different sources, a PowerPatch tape, an EXPRESS release (a **tape set** consisting of a Hewlett-Packard supplied SUBSYS tape and PowerPatch tape), or a combination of PowerPatch tape with reactive patches.
- Provides the added flexibility of allowing customization of a subset of the patches through its online **VETO** and **FORCE** functions, once the tape sets have been selected.

Phase II

Phase II does the following:

- Updates from tape (if needed)
- Re-runs HP Patch/iX to complete the installation by restoring all patched STORE files, streaming any patch installation jobs, and cleaning up its work environment.
- Uses a simple AUTOINST-style interface and requires a dedicated system.

Customizing Patches

Along with the new HP Patch/iX tool new patch information files are introduced that are delivered automatically with each patch. Each reactive patch is delivered with a binary file that is used by HP Patch/iX to evaluate the patch against the target system, and an ASCII file that is read by HP Patch/iX to display detailed information to you through its viewers.

The ASCII file is provided to aid in selection of patches to either be vetoed from installation or forced on the system. Each PowerPatch tape is delivered with a set of information files analogous to the reactive patch information files. However, unlike reactive patches, each PowerPatch tape is delivered with a set of KSAM files to be used by HP Patch/iX (just as it is today), and one comprehensive ASCII file for all patches on the tape containing the extended information.

The template for patch information contained in the ASCII file is as follows:

Patch ID	Seven-character patch name and one-character version letter.
One-line Description	70-character field that summarizes the purpose of the patch.
General Release Text	Text that describes the cause and fix text for the patch, any external changes experienced once the patch is applied, and hardware and software dependencies associated with the patch.
Special Instructions Text	Text that describes any special instructions that need to be followed in order to complete the patch installation beyond the normal patching procedures.
Product Numbers	Field that lists all of the products affected by the patch.
KPR Numbers	Text that provides the Known Problem Report (KPR) numbers that were fixed by the patch.
Patch Supersedes Tree	Text that provides a hierarchical view of the patches that were superseded by this patch.
Patch Component	List of files and library components that are included in the patch.

As part of the customization, it is often important to know which patches have been previously applied to the target system. HP Patch/iX provides a view of the patches previously installed on the system. It gives the following patch ID information:

- Origin of the patch (e.g., PowerPatch ID)
- Installation date
- Person who installed the patch

A second view provides a list of patches available for installation. The list of available patches may come from several sources, including PowerPatch tape, reactive patch tape(s), and electronic download. HP Patch/iX loads all of the information files to determine the set of available patches.

HP Patch/iX works similarly to the current PowerPatch process in that it evaluates each patch against the target environment through recognized checksums of each of the patch components, and availability of all other patches required by the patch (i.e., all patches required by the target patch must be available for installation at the same time, and must pass patch qualification criteria). HP Patch/iX applies the same qualification criteria to both PowerPatch patches and reactive patches.

Once the patches have been qualified, HP Patch/iX displays the list of patches sorted by Patch ID with a status of YES or NO in the QUALIFIED column of the display. At this point, you have the option of VETOing those undesired patches that QUALIFIED (denoted by YES in the column), or attempting to FORCE those patches that did not QUALIFY (denoted by NO) and requalifying the new subset of patches. When satisfied with the selection, you may opt to exit this screen and create the CSLT/STORE tape. Once the tape is created, Phase I of HP Patch/iX is complete.

Phase II of the process requires a dedicated system and must be performed at the system console. Because it requires a dedicated system, Phase II should be performed at your convenience (e.g., scheduled system downtime). Phase II of the process uses a simple AUTOINST-style interface that acknowledges the successful completion of Phase I, then asks you to continue via a yes/no prompt.

If you opt to continue, HP Patch/iX:

- Restores the patched STORE components from the tape.
- Streams all patch installation jobs.
- Cleans up the INSTALL.SYS work group.
- Terminates, completing the patching process.

Note

If an update is required (i.e., Phase I of HP Patch/iX created a CSLT/STORE tape, as opposed to a STORE only tape), you **must** perform the update before re-running Patch/iX. When HP Patch/iX is re-run, the tool determines if an update was required and whether it was performed. If it was not performed but is required, the tool does not continue with Phase II of the process.

DTS/TIO Dynamic Configuration and Host-Based Switching Technical Overview

by John Spitzer
Commercial Systems Division

Introduction

This article provides more detailed information on the new product features of the DTS/TIO introduced in the article, "DTS/TIO Dynamic Configuration and Host-Based Switching," in Chapter 6, "Networking/Client-Server."

This article provides detailed discussions of the following new features:

- Dynamic Configuration of DTS/TIO Devices
- Automatic Configuration of DTC Servers
- Shutdown and Restart of the DTS
- Terminal Switching on Host-Based DTC Ports
- Back-to-Back Configuration
- Domain Name Service and IP Routing
- Forced Data Forwarding
- Support of 1024 Profiles
- Capability to Specify Starting LDEV of Non-Nailed Pools

For complete information on the product features, see the manual, *Configuring System for Terminals, Printers, and Other Serial Devices* (32022-61000).

Dynamic Configuration of DTS/TIO Devices

Dynamic Configuration provides the capability to perform DTS/TIO configuration tasks without the need to shutdown the HP 3000 system. Most configuration tasks can be performed without affecting other TIO users on the system.

Configuration Tasks

The following summary lists the configuration tasks that can be done using Dynamic Configuration:

- Add or delete one or more LDEVs.
- Add or delete cards in a DTC.
- Add or delete entire DTCs.
- Add or delete non-nailed TIO or PAD LDEVs.
- Add or delete class names in a terminal or printer profile.
- Change most fields in a terminal or printer profile.

- Change many of the DTC configuration parameters such as SNMP parameters, timers and the type of logging that is enabled.

The following is a list of those tasks that **cannot** be performed dynamically. These changes require a DTS shutdown and restart or system reboot.

- Change from Host-based to PC-based.
- Change from PC-based to Host-based.
- Change from Host-based TIO only to Host-based X.25.
- Change from Host-based X.25 to Host-based TIO only.
- Change the LINK name and physical path of LANIC.
- Change Native Language I/O (NLIO) from Y to N.
- Change the DTC name.
- Change the DTC node name.
- Change the DTC LAN address

The last three DTC changes can be accomplished by deleting the existing DTC and adding the configuration for a replacement.

To make Dynamic Configuration changes, use **NMMGR** to change the configuration file NMCONFIG.PUB.SYS, then invoke the command **DTCCNTRL**. **DTCCNTRL** can be invoked by **NMMGR** after DTS validation completes, or later by you from the CI prompt.

Each time the (Save Data) key is pressed on any DTS configuration screen, **NMMGR** records the name of the screen that has changed in a special record in the NMCONFIG file. This special record is used to guide **DTCCNTRL** to make the desired changes.

DTCCNTRL determines the changes made to the configuration file and sends necessary messages to other DTS managers to cause them to change the DTS/TIO configuration. In all cases, **DTCCNTRL** obtains status from the DataComm Configurator (DCC) to determine that DCC Startup was OK.

- If **DTCCNTRL** determines that the change affects the characteristics of the DTC, then it sends messages to the DTC Manager (DTCM) to update the configuration of the DTC.
- If the desired changes affect LDEVs on the host, then messages are sent to the DataComm Configurator (DCC) to implement the change. DCC then sends the appropriate message or calls the OS functions to make the changes.

NMMGR can keep up to 70 changes. This means that you can change a maximum of 70 different screens before you invoke **DTCCNTRL**. After the 64th change, **NMMGR** displays a warning that tells you the limit is near. If more than 70 changes are made, **NMMGR** warns that the current change will not take effect using **DTCCNTRL**. Those changes made after 70 entries are not kept and cannot be made dynamically.

After `DTCCNTRL` is invoked and the changes are successful. `NMMGR` can again keep up to 70 new changes. Those changes that were not successful are saved so they can be retried at a later time. When a new DTC is added. `NMMGR` considers this to be one change even if you save data on many screens to configure this DTC.

All operations that delete LDEVs are non-destructive. This means that if an LDEV to be deleted is in use, then the delete is not performed. The operator must take the appropriate action to free the device. For example, they must stop the spooler on a spooled serial printer or log off the session on a terminal LDEV. If multiple LDEVs are deleted, then only free LDEVs are deleted. Those LDEVs that were not free are processed again the next time `DTCCNTRL` is invoked.

Some changes may affect more than one user. Usually this may happen when the change is to a DTC characteristic such as adding a card. This is because the DTC needs to be powered off or reset. Users on other DTCs are not affected.

Some changes require additional steps to be completed once `DTCCNTRL` has made the desired changes. These possible additional steps are:

- Logoff any sessions or close any jobs that access the changed devices.
- Stop and restart the spooler if the device is a printer.
- Stop and then start X.25/PADSUP.
- Reboot or reset the DTC.

`DTCCNTRL` displays a message when these steps are necessary and tells you which DTCs are affected.

NMCONFIG and NMCONFIX

The design of Dynamic Configuration has changed the way NMCONFIG.PUB.SYS and NMCONFIX.PUB.SYS are used by the DTS. On previous releases, the NMCONFIX file contained the configuration of the system from the last time it was booted with the ISL command `start norecovery`.

With MPE/iX 5.5, this file now reflects the actual DTS configuration of the system. `DTCCNTRL` determines what changes have been made to NMCONFIG.PUB.SYS by comparing it to NMCONFIX.PUB.SYS. When dynamic configuration operations are done, `DTCCNTRL` copies the configuration changes from NMCONFIG to NMCONFIX. This design imposes a restriction on the usage of NMCONFIG.PUB.SYS and NMCONFIX.PUB.SYS.

Note If any configuration file is copied to either or both NMCONFIG.PUB.SYS or NMCONFIX.PUB.SYS, then dynamic configuration is disabled until a DTS shutdown and restart is done, or the system is rebooted.

Dynamic configuration is disabled even if the files are backup files from the same system or other configuration files that are currently on the system. This is necessary because dynamic configuration compares the two files to determine the desired changes to be made and would behave unpredictably if any unexpected differences are recognized or if NMCONFIX.PUB.SYS does not match the actual DTS configuration.

Every time dynamic configuration is invoked DTCCNTRL checks an identification number stored in both NMCONFIG.PUB.SYS and NMCONFIX.PUB.SYS. If the numbers match and if they are the same as the number stored in DCC, then the dynamic changes are made. When the changes have completed, the number is updated. If the numbers don't match then the file that contains the mismatched number is considered to be a foreign file.

LOGDCC and LOGDCCBK

To increase supportability of dynamic configuration, DTCCNTRL logs to the LOGDCC file detailed information of what dynamic operations were done. This file contains a history of all changes done since the last DTS shutdown and restart or system reboot. If LOGDCC should become full, DTCCNTRL renames it to LOGDCCBK and creates a new LOGDCC file. When the system is rebooted or a DTS shutdown and restart is done, both files are purged and a new LOGDCC built.

Automatic Configuration of DTC servers

If you are using Host-based management to manage your DTCs, you can now add and configure a new DTC automatically without the need to specify detailed configuration information such as DTC cards/boards or LDEVs. You can use automatic DTC Configuration either within the NMMGR utility or outside NMMGR.

Automatic DTC configuration inside NMMGR is accessed from the DTC selection screen. Automatic DTC configuration outside NMMGR is done via the DTCCNTRL command which requires you to have Network Manager (NM) capability.

To perform an automatic configuration, you need to provide only the DTC's LAN address. A default DTC name and node name are generated for you to use. You can also specify your own names. You can optionally specify an IP address.

The automatic configuration process determines the number and types of cards in the DTC and creates the appropriate configuration in the NMCONFIG.PUB.SYS file. All serial cards are configured with all ports as nailed, direct connect terminals using the terminal profile TR10AUTO. All X.25 cards are assigned a default configuration that must be modified with correct Level 1, 2 and 3 information before X.25 or PAD can be started successfully.

This feature is only available when the HP 3000 is using Host-based DTC management. If the DTC is managed using PC-based management, automatic configuration is done from the OpenView Windows workstation running OpenView DTC Manager.

Shutdown and Restart of the DTS

DTS shutdown allows System Managers (SM) to shutdown the entire DTS/TIO subsystem without rebooting the HP 3000. This function deletes all LDEVs created by DTS and releases all system resources used by the DTS/TIO subsystem.

DTS Restart

DTS restart recreates the entire DTS/TIO subsystem using the current contents of the NMCONFIG.PUB.SYS file. With this capability you can resolve any DataComm Configurator (DCC) errors that may have occurred during system reboot or perform major DTS configuration changes such as changing from Host-based to PC-based management. To correct any DCC errors that occurred on reboot, you must correct the cause of the errors before performing a DTS restart.

DTS Shutdown

When DTS shutdown is invoked, all TIO LDEVs are scanned. If any LDEVs are in use, then the shutdown process is aborted. No users are affected. DTS shutdown can be invoked with an optional `forced` option. If the `forced` option is used, then all active LDEVs are reset and the spooler stopped on all spooled serial printers. This causes all sessions using the terminal ports to be aborted.

Terminal Switching on Host-Based DTC Ports.

Host-based terminal switching allows you to connect to other systems on the LAN from a terminal connected to a DTC that is using Host-based management. Previously, this capability was only available if the DTC was using PC-based management. Connections to other systems are restricted to one session or connection per DTC port. Connections from PAD ports are still restricted to the system that is managing the DTC. To access other systems, you must log off or close the connection before connecting to a second system. This feature also allows serial printers to be shared by more than one HP 3000 host.

For Host-based switching, the HP 3000 DTC managing system configures and downloads the DTC and must be configured to use Host-based management in NMMGR. If an HP 3000 system does not manage any DTCs, it can be configured as either PC-based or Host-based.

- If Host-based management is configured, then all port access must be through nailed LDEVs. These nailed LDEVs are created by configuring a fictitious DTC that is configured using a dummy LAN address.

Note The dummy LAN address must be 00-00-00-00-00-00.

- If PC-based management is configured then access can be made through either nailed or non-nailed LDEVs.

This feature is configured by using `NMMGR` to access a new screen in a profile that contains a field to enable switching. When terminal switching is enabled, you see a DTC User Interface prompt after the first carriage return is sent from a terminal connected to the DTC. From the DTC User Interface, you can enter a command to connect to a system and specify the name of other systems on the LAN.

The DTC tries to connect to this node using NS probe or TCP/IP protocols. If this DTC has Domain Name Services configured, then the DTC can use DNS to resolve an entered name into an IP address. See the section below that describes the new DNS configuration capability.

Back-to-Back Configuration.

MPE/iX Release 5.5 now provides the capability to configure` Host-based DTCs to use back-to-back connections, also known as extended switching. Back-to-back allows a connection to be established from one DTC port to another that is configured as a HOST port. This HOST port could be connected to another system such as an MPE V or other system through a serial connection. This allows a DTC terminal user to connect to the MPE V system. This connection can also be used to allow printers connected to DTCs to be sent output from the other MPE V system.

This configuration is done using `NMMGR` on the system managing the DTC. A new profile type, HOST, is provided. When this profile is assigned to a DTC port, the special LDEV number (-1) is used to tell the DTS not to create an LDEV on the HP 3000. The number of HOST ports created are not counted in the total number of LDEVs allowed on the HP 3000.

Domain Name Service and IP Routing

Host-based DTCs can now be configured to use Domain Name Services (DNS) for IP address resolution and to use IP routing. These services are configured when back-to-back connections through routers are used or connections are to be made from DTC ports to HP 9000s or third-party ARPA nodes. Domain Name Services allows you to use DNS to resolve system names into IP addresses when switched DTC ports are connected to ARPA (TCP/IP) nodes. IP routing is only needed when routers are used on the network.

This configuration is made using **NMMGR** and accessed from the DTC configuration screen. This screen allows you to specify the address of the DNS server, default local domain, router address and subnet mask.

Forced Data Forwarding

When PAD terminals are created, one of the features that can be configured is a parameter that is sent to the PAD port that determines the behavior of the PAD when it sends (or forwards) data to the HP 3000 host.

With MPE/iX Release 5.5, you can now use **NMMGR** to configure an additional data forwarding parameter in the Host-based PAD profiles to set the **Forced Data Forwarding** bit. When this feature is enabled in **NMMGR** for PAD terminals, the data forwarding parameters that are set at connection time are always used. If this feature is not enabled, then the data forwarding parameters would change depending on whether the session is at a normal read, is in VPlus mode, or has an Alternate End Of Record (AEOR) character set.

Prior to MPE/iX Release 5.5, this feature could only be set on the OpenView DTC Workstation when the system was PC-based or by specifying the appropriate profile in the call request from the terminal connected to the PAD. Host-base X.25 and PAD features are available if you have purchased and installed X.25 iX System Access and DTC/X.25 MPE/iX Network Link.

Support of 1024 Profiles

Profiles are used to define the characteristics of DTS serial devices. They are created using **NMMGR**. Each LDEV configured on the DTS is assigned a profile. With MPE/iX Release 5.5, the number of supported profiles is increased to 1024 (from 256 prior to 5.5).

Specify Starting LDEV of Non-Nailed Pools.

When non-nailed LDEVs are created, the system searches the configured devices starting at the lowest LDEV number and assigns the new non-nailed LDEV the first unconfigured LDEV number found. On MPE/iX Release 5.5 you can now specify the LDEV number where the search starts. The system now assigns the new non-nailed LDEV the first unconfigured LDEV number equal to or greater that the specified starting number. A different starting number can be set for PAD and TIO non-nailed LDEVs.

This starting number is dynamically configurable with the following restrictions:

- When the starting number is changed using **NMMGR**, there can be no non-nailed devices currently configured
- All non-nailed devices must be deleted before the new starting number takes affect.

Network Printing Technical Overview

by Steve Bitondo and Shelley Nelson
Commercial Systems Division

With Release 5.5 of MPE/iX, the Native Mode Spooler (NMS) now supports access to *network printers* on the HP 3000, that is, any Printer Command Language (PCL)-based printer that is attached to the HP 3000 via a TCP/IP network connection and a JetDirect interface card. The complete description of configuring and operating network printers is available in the revised edition of the *Native Mode Spooler Reference Manual* (32650-90166). This Communicator article provides a basic technical overview of network printing on the HP 3000.

This article is addressed to the system manager because it is that person who is primarily concerned with incorporating network printers into the data processing environment. The system manager oversees the printer installation and makes the necessary configuration changes to accommodate network printing. Installing the printers is not explained in this article or in the NMS Reference Manual. Configuring network printers is, and it includes the following tasks:

- Adding the printers to SYSGEN's I/O configuration
- Creating the network printer configuration file, NPCONFIG, in the PUB group of the SYS account.
- Creating and using setup files
- Modifying XL.PUB.SYS for network printing
- Using the same setup file for serial and network printers

Working with setup files may be handled by any users (not just system managers) who want to more closely control how their individual spool files are printed.

Configuring a Network Printer with SYSGEN

To use SYSGEN to add a network printer to your system's I/O configuration, do the following:

1. Make sure you are logged on as MANAGER.SYS and run SYSGEN. At the CI prompt, enter:

   ```
   :run sysgen.pub.sys
   ```

2. At the sysgen prompt (>), enter `io` to start the I/O configurator.
3. Define the logical device identification for each network printer that you want to add. To use the default configuration values, specify `HPTCPJD` as the device identification and designate the path as `NONE`. For example, to configure LDEV 19 as a network printer, enter:

   ```
   io> ad ldev=19;id=HPTCPJD;path=NONE
   ```

To view the device configuration, enter the **ld** command, for example:

```
io> ld 19

  LDEV:      19 DEVNAME:                OUTDEV:        0  MODE:          OS
    ID: HPTCPJD                         RSIZE:        66  DEVTYPE: PP
  PATH: NONE                            MPETYPE:      32  MPESUBTYPE:  0
 CLASS: NETLP
```

4. Enter the **hold** command to save the modified I/O configuration, and then type **exit** to leave the I/O configurator.
5. At the SYSGEN prompt, enter the **keep** command and then type **exit** to leave SYSGEN.
6. At a convenient time, perform an orderly shutdown of the system and then restart it to have the new I/O configuration take effect.

Creating the Network Printer Configuration File

The network printer configuration file NPCONFIG.PUB.SYS is a flat ASCII file that the system manager creates and modifies using a text editor. The purpose of the NPCONFIG file is to supply to the system additional configuration data about network printers that is not defined in SYSGEN. The NPCONFIG configuration file is designed to be extensible. As needed, for example, when placing a new network printer in service, the system manager may update the entries in NPCONFIG.

At a minimum, NPCONFIG must have the following information:

- An LDEV-specific entry for each network printer.
- The printer's network (IP) address or a domain name which the spooler can resolve to an IP address.

For example, a complete though minimal entry in NPCONFIG for the network printer designated as LDEV 19 might be:

```
19      (network_address = 192.187.63.25)
```

Most NPCONFIG files will also have one global entry whose items are applicable to all network printers. This makes it more convenient to configure a group of printers that belong to the same "family" since it is unnecessary to repeat identical configuration items for each LDEV entry. (With one exception, `setup_file`, items found in the LDEV specific entries take precedence over those in the global entry.) For example, a global entry for a group of LaserJet 4Si printers might look like this:

```
global (setup_file = LJ4SISET.HPENV.SYS      # LaserJet 4Si setup file.
        message_interval = 60                # Repeat msgs every >= 60 secs.
        banner_source = 1                    # Upper tray has colored banner
                                             #   paper.
        banner_trailer = FALSE               # Only need a header page.
        pjl_supported = TRUE                 # LJ4Si is a full PJL device.
        jam_recovery = TRUE)                 # Reprints jammed pages by itself.
```

Any text that follows the pound sign (**#**) are comments, and can help make the NPCONFIG file self-documenting. It isn't necessary to enter values for every possible item in either a global or LDEV-specific entry. Items that you do not specify automatically assume the default values.

NPCONFIG File Security

The system manager creates the NPCONFIG.PUB.SYS with the same security matrix as a typical file in the PUB group of the SYS account. As a result, only users with SM capability or a user logged on in PUB.SYS can make changes to the file or to its security matrix. If you plan to add an ACD to the file, do so **only** after careful consideration of its impact.

Items in an NPCONFIG entry

The table on the next page briefly describes each of the items used in the NPCONFIG file and lists the default value.

Summary of NPCONFIG File Items

Item	Definition
network_address	Network address of the printer, specified either as an IP address or as a domain name that resolves to an IP address. This item is required. No default.
TCP_port_number	Port number used by the spooler to establish a TCP connection to the target printer. Default = 9100.
program_file	Name of the output spooler program file invoked for the network printer. Default is OUTSPTJ.PUB.SYS.
poll_interval	Initial time interval, in seconds, that the spooler waits to retry connecting to the printer. Default is 10 seconds.
poll_interval_max	Maximum amount of time, in seconds, that the spooler waits to retry connecting to the printer. Default is **poll_interval**.
setup_file	Fully-qualified MPE/iX name of a file containing information for a printer setup string. No default.
run_priority	Scheduling queue assigned to the output spooler process. Default is CS.
SNMP_get_community_name	The "SNMP Get Community Name" for this printer, which determines who can check printer status. Default is ALL.
data_timeout	Amount of time, in seconds, that the spooler waits for a specific network I/O request to complete before verifying that the printer and network are functioning correctly. Default is 10 seconds.
snmp_timeout	Amount of time, in seconds, that the spooler waits for a printer status check to complete. Default is 5 seconds.
snmp_max_retries	Number of times that the spooler should cycle through printer status checking before consulting **message_interval** to display error/warning messages. Default is 3.
message_interval	Minimum interval, in seconds, at which a printer status message is redisplayed, as long as it applies. Default is 0, which displays the message only once.
banner_intray	Sends a paper source command to the printer to allow banner pages to be taken from a separate paper tray. No default.
data_intray	Sends a paper source command to the printer which specifies the paper tray for normal data pages. No default.
banner_header and **banner_trailer**	Determines whether or not spool files are printed with a header, a trailer, both, or neither. Default is TRUE. The CI command **HEADOFF** overrides these settings.
pjl_supported	Specifies whether or not the spooler tests a printer to see if it can effect Page Level Recovery and actual Page Count Logging. No default.
jam_recovery	Specifies whether or not the spooler invokes its own jam recovery procedure. Default is FALSE.
socket_trace	When enabled, initiates a socket-level trace of the TCP connection socket. Default is OFF.

Making changes to NPCONFIG

The spooler process for each configured network printer reads the NPCONFIG file once, briefly, when it first starts executing. This occurs when the system first boots if the LDEV is configured as `OS` or "output spooled" in SYSGEN. The spooler also reads the NPCONFIG file whenever a user issues a `SPOOLER` *nn*`;START` or a `STARTSPOOL` *nn* for the printer.

You may edit the NPCONFIG file at any time. The changes will take effect on a particular LDEV the next time that the spooler process starts for that printer. The changes will have no effect on a spooler process that is already executing.

You create the NPCONFIG.PUB.SYS with the same security matrix as a typical file in the PUB group of the SYS account. As a result, only users with SM capability or a user logged on in PUB.SYS can make changes to the file or to its security matrix. If you plan to add an ACD to the file, do so **only** after careful consideration of its impact.

Using Setup Files with Network Printers

Any user, regardless of their assigned capability, may create files containing customized setup strings (analogous to environment files) to specify the printer operating mode for network printers. Such files can be used in one of two ways: If you have system manager (SM) capability and therefore can edit NPCONFIG, you can name such a file as the **`setup_file`** for a particular printer LDEV so that it becomes the default. For example, the following entry in NPCONFIG specifies the setup file LJ4SISET.HPENV.SYS for LDEV 19:

```
19      (network_address = 192.187.63.25
         setup_file = LJ4SISET.HPENV.SYS)
```

If you do not have SM capability, you can create a setup file and direct the spooler to use it while printing a specific spool file by entering the **`ENV`** parameter in the `FILE` command or the (HP)FOPEN intrinsic. For example:

```
:FILE MYOUT;DEV=NETPRINT;ENV=PCLELITD
:FCOPY FROM=MYFILE;TO=*MYOUT
```

When you specify a setup file in an **`ENV`** statement, the spooler assumes that it defines the entire printing environment. As a result, it supersedes all other setup file specifications in the file NPCONFIG.PUB.SYS, which the system manager created for your network environment.

The MPE spooler expects the contents of setup files to be the raw data stream needed by the printer. Comments are not allowed. You can create either a byte-stream file or a record-oriented file. A byte-stream file is sent as-is to the printer. For record-oriented files, the spooler deletes any carriage control (CCTL) code, then trims leading and trailing blanks and concatenates records to arrive at the sequence sent to the printer.

If your printer supports PJL, the resulting data stream can include both PCL and PJL sequences, but they should be ordered so as to make sense to the printer.

Caution

Users are entirely responsible for the contents of setup files and their resulting effect on operation. The spooler does not interpret or alter them in any way (except to remove CCTL and blanks as described in this section). In particular, you should be familiar with PCL and PJL concepts as well as the features of your target printer before attempting to create a setup file.

A "setup", or "setup string", is that sequence of (raw) PCL and/or PJL commands used to place the target printer in a specific operating mode. For example, the PCL sequence Esc&*l*0O selects portrait mode, while Esc&*l*1S selects duplex (long edge binding) operation.

Setup file hierarchy

There are four hierarchical levels of setup available to network printer users:

1. An ENV file specification issued via the **ENV=<filename>** keyword in a FILE equation or in an (HP)FOPEN intrinsic. If you use the **ENV** statement, it is applied at the time the file is opened and **it supersedes all other setup file specifications** described below.
2. A global setup file specification in NPCONFIG.PUB.SYS, which applies to all network printers and is applied at print time.
3. An LDEV-specific setup file in NPCONFIG.PUB.SYS, which is applied at print time. Its contents are appended to any global setup file specification, adding to or overriding the global print setup information.
4. The default MPE/iX environment, which you cannot modify.

Accessing Network Printers

A traditional spooler process manages a channel-attached printer or a serial line printer connected via a DTC. Once the printer has been allocated to that process, it is the exclusive property of the process until the process terminates. A network spooler process manages the connections to a network printer. This printer is *not* the exclusive property of the process, but must be shared among an unknown number of hosts. The network spooler process responds to this need by competing with all other hosts to connect to the printer whenever it has output to print. Whenever it fails because another host has attached the printer, it retries at configurable intervals. (See the `poll_interval` and `poll_interval_max` item descriptions in the NPCONFIG section later in this chapter.) The network spooler process releases the connection to the printer after each copy of each file, thus allowing other hosts a chance to access the printer.

Note that although the network spooler process releases the *printer connection* between files, it does not release or surrender the MPE LDEV. The LDEV belongs exclusively to the network spooler process, and its spool queues remain open.

Other than the competition among hosts to secure a network printer, there is no other operational difference between a network printer and a channel attached printer. You control spooler processes for both with the same commands and syntax. Thus, `STARTSPOOL`, `SPOOLER <ldev>; START`, `STOPSPOOL`, `SPOOLER <ldev>; SUSPEND` all work for both types of printers. Refer to Chapter 5 in the *Native Mode Spooler Reference Manual* (32650-90166) for detailed information about these commands.

Modifying XL.PUB.SYS for Network Printing

All user application programs and networking programs resolve external procedure references through XL.PUB.SYS, thus leaving it open. The new TCP/IP network printer spooler process, OUTSPTJ.PUB.SYS, also resolves externals through XL.PUB.SYS, which means that XL.PUB.SYS must be open to support network printing.

Many users and a number of vendor software packages run the MPE/iX Link Editor directly on XL.PUB.SYS to install new code modules. To do this, no application programs can be running and the network must be stopped. Since this may be unacceptable to your user community, there is an alternative: modifying a copy of XL.PUB.SYS with the Link Editor and pointing SYSGEN to the new copy.

You may choose either the direct or the indirect method to modify XL.PUB.SYS; both are explained below.

To directly modify XL.PUB.SYS for network printing, follow these steps:

1. Unload XL.PUB.SYS.
2. End any application programs that are running.

3. Shut down all networking which includes issuing the `NSCONTROL STOP` and `NETCONTROL STOP` commands.
4. Stop the spooler(s) on all of your network printer LDEVs by issuing the `STOPSPOOL <ldev>` or `SPOOLER <ldev>;STOP` command.

To make a copy of XL.PUB.SYS and modify it, follow these steps:

1. Copy XL.PUB.SYS.
2. Modify the copy with the Link Editor.
3. Run SYSGEN.
4. Point SYSGEN to the new copy of XL.PUB.SYS.
5. Make a boot tape.
6. At your earliest convenience, update the system from the new boot tape.

Using the Same Setup File for Serial and Network Printers

Many customers will want to migrate from serially-connected LaserJet series printers to TCP/IP network printers. This migration may not happen all at once, so for awhile, there would be a mix of both kinds of printers. This section describes a transparent method by which output may be directed to either kind of printer with identical results.

Serial printers often use a TTUTIL file to define their printer setup string (`LTHD` in the example below). Users invoke it with an MPE file equation, for example:

```
:FILE LTRHEAD;DEV=SERIALLP;ENV=LTHD.TT.SYS
```

However, TTUTIL files cannot be used with network printers. If the above file equation is used, the `ENV` will be ignored.

Most of the settings in a TT file deal with terminal configuration. They are irrelevant to a serially-connected printer. It is enough to preserve the PCL codes in the initialization string, as shown in the "VFC And Initialization" screen of `TTUTIL.PUB.SYS`. Note that it doesn't matter how the initialization string gets to the printer, just as long as it does get there.

Although a serial printer TT file cannot be used (is ignored) by a network printer, a network printer setup file invoked by the `ENV` keyword of a file equation will work with either printer. This is because the contents of such a file are written directly to the output spoolfile as normal write records, and they precede all user data.

Therefore, all that is necessary is to place the contents of the initialization string of your TT file in a network printing setup file (an ordinary user file), and either modify the above file equation to specify this new file as your `ENV` file, as in the following example:

```
:FILE LTRHEAD;DEV=SERIALLP;ENV=NPSETUP.PUB.SYS
```

or, replace the TTUTIL file `LTHD.TT.SYS` with your new NP setup file, calling *it* `LTHD.TT.SYS`. If you use this second method, it is not necessary to change any job streams or scripts at all.

Telnet/iX Server Functionality Details

by Lyn Hirsch and Cas Caswell
Commercial Systems Division

Introduction

This article provides more detailed information about the Telnet/iX Server functionality introduced in the article, "Introducing the Telnet/iX Server," in Chapter 8, "Networking/Client-Server." This information applies to application managers/developers and end-users.

The Telnet/iX Server is available in two parts:

- With the MPE/iX 5.5 Release, the Telnet/iX Server will allow a user to logon to the HP 3000 and use all MPE/iX CI commands, as well as issue some Telnet client commands to the Server, which are described in the "Telnet Client Commands Support" section later on in this article.
- Full functionality, as documented for the Telnet/iX Server product in the 5.5 version of the following manuals, will be available in the Fall 1996:
  - □ *HP Telnet/iX User's Guide* (36957-90154)
  - □ *Configuring and Managing MPE/iX Internet Services* (32650-90835)
  - □ *Asynchronous Serial Communications Programmer's Reference Manual* (32022-61001)

The following information describes the restricted functionality that is available with this software.

FCONTROLs and FDEVICECONTROLs Support

The chart below shows the list of currently supported FControls (FC) and FDevicecontrols (FDC) on a Telnet/iX session. Applications that depend on functionality provided by FC or FDC intrinsics which are not listed on this chart, may not work properly.

FDC	FC	Description
4	12,13	Set echo at a terminal
14	34,35	Set line deletion response
30,32	N/A	Define read trigger character
36	N/A	Define backspace character
41	N/A	Define subsystem break character
55	N/A	Set backspace response action
57	N/A	Obtain subsystem break character

Important Details Please Read

- Note that since Fcontrols 4, 20, 21, 22 and FDC 2, 7, 8 are not on the supported list, both the MPE logon and password prompts will not time out.
- VPLUS and user block modes and binary mode are not supported at this time.
- Typeahead is not supported. All data entered before a read has been posted, including the events break , subsystem break , XON, and XOFF, will be discarded.

Flow Control of Screen Display

When an application is sending large amounts of data to the screen, you can pause the display to take note of a particular line. To do this on most terminal emulators, use the STOP key to introduce a local flow control. That is, the terminal emulator stops displaying data to the screen, but does not communicate the stop to the Telnet/iX Server system.

Note

Using the MPE traditional key strokes of CTRL-S, XOFF, and CTRL-Q, XON, will not work. The Telnet/iX Server driver will either discard the flow control characters, if no read is posted, or pass the flow control characters to the user's application as data if there is a read posted. In either case, the flow control characters will not be interpreted in the manner the user intended.

BREAK Key on a Keyboard

If you type BREAK using the BREAK key from a keyboard, the Telnet/iX Server driver may **not** recognize the break event. It depends on what the Telnet client maps the BREAK to.

For example, one Telnet client maps the BREAK key to a `NULL` character and sends it to the Telnet/iX Server. Since the `NULL` character on its own has no meaning in the Telnet protocol, the `NULL` is passed to the application as data and not recognized by the Telnet/iX Server as a break event.

To generate a break event from a Telnet client, you can do one of the following:

1. Use the **`send break`** command from the Telnet client.
2. Use the defined value for the *quit* variable from the Telnet client. You can find out the value for the *quit* variable by using the display command at the Telnet client prompt (the default value for the *quit* variable on HPUX is ^\). Note that this is not the same *quit* as the Telnet client quit command. The client must be recognizing control characters for this to work.

   For example, from an MPE session established with Telnet from an HPUX system:

```
: ^]                        To get to the Telnet prompt.

telnet> toggle localchars
Will recognize certain control characters.

<CR>                        Press Carriage Return for the prompt.

:
```

From this point on, typing the value for the *quit* variable from the Telnet client (e.g., ^\) causes a Telnet **send break** command to be sent to the Telnet/iX Server.

3. Some Telnet clients allow you to configure what the BREAK key will send to the Telnet server. Using the Telnet/iX Server, configure the client to send the Telnet command **send break**, when the user presses the BREAK key.

Telnet Client Commands Support

A user may enter the user interface portion of their client to use some of the commands provided there. These user interface commands will fall into two families:

- The first family of commands are those that are purely local to the client, such as display or status.

  The Telnet/iX Server is not aware of the first family of commands and will not be affected by them.

- The second family of commands are those that will communicate with the Telnet/iX Server, such as `open` or `close`. However, not all of the second family of commands are supported at this time, particularly the toggle option commands.

  The commands that are supported are:

  ```
  open
  send break
  send AYT
  send IP
  send EC
  send EL
  quit
  close
  ```

It is important to remember the following translations that will occur with the Telnet/iX Server:

`send IP`	Mapped to subsystem break on MPE.
`send break`	Mapped to system break on MPE.
`send EC`	Mapped to the currently defined backspace character.
`send EL`	Mapped to line delete.

Maximum Number of Concurrent Sessions

The current maximum number of concurrent sessions using the Telnet/iX Server to connect to an MPE host is 250.

TLNETDOC.ARPA.SYS

The file TLNETDOC.ARPA.SYS has not yet been updated with information for the Telnet/iX Server.

UPS Required for Support of New SCSI Disks

by Jeff Flowers
Worldwide Customer Support Operations

Introduction

Recent changes in disk drive characteristics have forced a change in the HP 3000 powerfail recovery strategy. With the 5.0 Express 3 and MPE/iX 5.5 introduction of new SCSI disk drives (introduced in the article, "New Disks for MPE/iX 5.5," in Chapter 9, "Peripherals") the HP 3000 requires UPS protection for the listed disk drives regardless of how these disks are connected to the system.

Note

9x8, 9x9KS, and 99x SPUs with these drives embedded may take advantage of the UPS already shipped with the system if sufficient capacity is available. All other HP3000 models will require external UPS capacity to cover the external versions of the drives.

The MPE/iX 5.5 Release includes enhancements that allow the host system to perform an automatic protective shutdown when the level of power in the HP PowerTrust UPS gets dangerously low. These enhancements are available for MPE/iX 5.0 as patch MPEHXW6 or as part of PowerPatch C.50.05. The need for this powerfail requirements change is being driven by changes in the way disk drives are being designed.

Background

To ensure data integrity of the system, we must ensure that all writes issued from the host are completely written to disk media prior to a power outage. If the power failure is in danger of exceeding the time limit of the UPS, the SPU must terminate any further transactions to the I/O subsystem. The UPS requirement protects the system from the loss of AC power while the protective system abort software protects against UPS exhaustion.

Recommendation

We strongly recommend that customers use the HP PowerTrust UPS and the automatic protective shutdown software available on PowerPatch 5 (C.50.05) based on MPE/iX General Release 5.0 and in the MPE/iX 5.5 Release. Only an HP PowerTrust is able to communicate to the host the level of power protection remaining in the UPS. When this level reaches a minimal threshold, the HP3000 will issue a protective system abort to halt all transactions to the I/O system thus preventing the possibility of data corruption. Without the HP PowerTrust UPS, the responsibility for a manual shutdown of the system would fall to the operator. Please see the associated technical article, "Protective System Abort #5300 from UPS Monitor/iX," in this chapter for more information.

9x7 System Considerations

Protecting your 9x7 SPU and embedded disk drives with a UPS has not been certified for powerfail protection and therefore is not currently supported. 9x7 systems should not use the new disk drives embedded with the SPU (see the table in the "New Disks for MPE/iX 5.5" article in Chapter 9, "Peripherals"), instead. 9x7 systems should use the older A3087A for the embedded 2GB disk drive and the A3182A for the embedded 1GB disk drive. The Battery Backup Unit in the 9x7 will protect main memory and these embedded drives are able to protect themselves as they do not have the newer drive characteristics.

The 9x7 systems can use the new release disk drives in external cabinets (listed in the "New Disks for MPE/iX 5.5" article in Chapter 9), which require:

- The Protective System Abort enhancement described in the article, "Protective System Abort #5300 from UPS Monitor/iX," in this chapter.
- An HP PowerTrust UPS for external disk drives.

The HP PowerTrust UPS can be connected to the 9x7 system for host communication through the DTC. Host connection via the DTC will enable the protective shutdown in the event of UPS exhaustion for external disk drives. An example of this would be if a power outage occurred on external disk drives but did not affect the SPU.

HP PowerTrust UPS Advantages

Only the HP PowerTrust UPS is able to communicate with the host system via a serial (RS-232) link. In this way, the host can monitor the power remaining in the UPS and initiate the automatic protective system abort. Third-party UPSs and plug-level protection methods do not communicate with the host system. If you do not use an HP PowerTrust UPS, you must have adequate powerfail protection and procedures in place to ensure that an operator is performing the functions of monitoring the UPS power and shutting down the system if UPS failure is imminent.

Why Have a Protective System Abort?

Without the protective system abort, there is a small probability that undetected data corruption could occur. To eliminate the risk of corruption, a reload would need to be performed in order to return the system to a known good state. If a powerfail does occur prior to a system shutdown, we recommend that the customer contact their support organization to help determine if a reload is needed.

Further Information

- For information about the protective system shutdown for the UPS Monitor/iX, see the related technical article, "Protective System Abort #5300 from UPS Monitor/iX," in this chapter.
- For part numbers of disks which require HP PowerTrust support, please refer to the "New Disks for MPE/iX 5.5" article in Chapter 9, "Peripherals.".
- For detailed information about the HP PowerTrust UPS, please refer to the article."Introducing the HP PowerTrust UPS Monitor/iX," in Chapter 3, "System Management" in the MPE/iX General Release 5.0 *Communicator*.

Protective System Abort #5300 from UPS Monitor/iX

by Jeff Flowers
Worldwide Customer Support Operations

Introduction

All of the disk drives that were released with MPE/iX Release 5.5 (or MPE/iX-Express 3 based on General Release 5.0 with PowerPatch) require a UPS for powerfail recovery. The disk drives and their model numbers, configuration IDs, and ordering part numbers are listed in the article, "New Disks for MPE/iX 5.5," in Chapter 9, "Peripherals."

MPE/iX 5.5 includes a new function in the UPS Monitor/iX software that causes UPS Monitor/iX to invoke a Protective System Abort #5300 when the UPS internal battery is drained to the "low battery charge" point. (Approximately two minutes of reserve power remains in the UPS.) This functionality is available for MPE/iX General Release 5.0 as patch MPEHXW6 or in the C.50.05 PowerPatch.

Background

The purpose of this Protective System Abort is to ensure that the system ceases to issue Write I/O requests to disks and allows the disks to complete their I/O Writes in the remaining 2 minutes of UPS power. Stopping Write I/O requests is required to guarantee the integrity of XM Log data and to ensure system recoverability once the system restarts on power resumption.

Some new disk drive models exhibit a behavior of out-of-order data write operations just at the point of power failure, leaving an incomplete XM Log entry. Although the probability is extremely low, this condition could occur during power failure. HP has chosen to enforce the integrity of XM data on a customer systems' disk storage using the tactic of a Protective System Abort and UPS support.

In practice, the Protective System Abort does not change the availability of the system, since the system is about to stop anyway due to a true loss of power. The HP PowerTrust UPS (with a fully charged battery) supplies power for 15 minutes after the loss of AC input power, and a "low battery charge" system warning is sent to the host after 13 of the 15 minutes of reserve power. Chances are good that if the AC input power has not been restored during this 13 minute period, it is not going to be restored within the final 2 minutes, and so the system components being powered by the UPS are going to lose all power very soon. Therefore, the stoppage of the system I/O via Protective System Abort #5300 with 2 minutes of power remaining does not materially reduce system availability, but it does protect disk data integrity.

When the UPS battery has drained down to the "low charge" level, the UPS Monitor/iX software takes the following actions:

1. When AC power first fails, UPS Monitor/iX warns the system operator with a console message, like this:

```
9:20/56/UPS LDEV 22 reports loss of AC input power. (UPSERR 0033)
```

2. After about 13 minutes (depending on how much power is being drawn from the UPS), the "low battery charge" condition occurs and the UPS Monitor/iX notifies the system operator, via a console message, that the system is about to be stopped.

```
9:33/56/UPS LDEV 22 reports Low Battery Charge condition. (UPSWRN 0037)
9:33/56/** WARNING ** System is about to be ABORTED by UPS Monitor
due to imminent loss of power from UPS with LOW BATTERY condition.
This intentional System Abort will safeguard the integrity of data
on disks. (UPSERR 0621)
```

3. Approximately five seconds after the warning message is issued, the system is aborted, and the following abort message appears on the system console:

```
SYSTEM ABORT 5300 FROM SUBSYSTEM 137
SECONDARY STATUS:  INFO = -37, SUBSYS = 137
SYSTEM HALT 7, $14B4
```

The recovery action is to get AC power restored, and then restart the system.

Important Details Please Read

The MPE/iX 5.5 enhancement or MPE/iX General Release 5.0 MPEHXW6 patch provides the system capability **only** for HP-supplied HP PowerTrust UPSs that are properly installed, cabled, and configured for monitoring by UPS Monitor/iX software. A third-party UPS cannot communicate to the UPS Monitor/iX and initiate the ordered Protective System Abort.

11

Product Release History

The following provides information on the currently supported Commercial Systems MPE/iX releases and products. Included are the MPE/iX release or SUBSYS VUF and a list of products introduced. It also provides information on significant changes made to a release.

MPE/iX Product Releases

Release	SUBSYS	Date Code	Product(s) Introduced/Added
B.40.00		R3217	MPE/iX Release 4.0 (Core Software Release)[1] Token Ring 3000/iX Network Link (HPJ2167A) NCS Runtime (HP36961A) HP 5000 Model F100 (HPC2753A) HP GlancePlus Pak (HPB2953A) Support of HP-FL disks/disk array devices: HPC2252B, HPC2252HA, HPC2254B, HPC2254HA
	B.40.02	R3337	MPE/iX Release 4.0 Express 1 IMAGE/SQL (HP36385) ALLBASE SQL ASM (HP30368) ALLBASE/NET (HP30604) ALLBASE/XL HPSQL (HP36216) HP Sockets/iX (HP92616) HP GlancePLUS (B1787B) Perf Collector XL (B1794B) Info. Access SQL/XL (ACCESSQL) Support of HP7980S and HP7980SX SCSI Tape Drives HP 3000 Series 980/400 HP 3000 Series 990DX, 992DX/200, 992DX/300, 992DX/400
	B.40.03		MPE/iX Release 4.0 Express 2 No new products

MPE/iX Product Releases (continued)

Release	SUBSYS	Date Code	Product(s) Introduced/Added
B.40.00	B.40.04	R3412	MPE/iX Release 4.0 Express 3
			NetWare for the HP 3000 (HP32020)
			HP/File Library (B3614A)
			HP Schedule (B3617A)
			OpenDESK Intrinsics (B3618A)
			HPDeskmon (B3619A)
			HP Open DeskManager (B3604A)
			HP Open DeskManagerPLUS (B3605A)
	B.40.05	R3427	MPE/iX Release 4.0 Express 4
			Transact/iX (HP30138)
C.45.00		R3249	MPE/iX Release 4.5 (Major Software Release)[2]
			Curses Library (B2477A)
			HP OpenView Console (B3118A)
			MPE/iX Developer's Kit (36340A)
X.50.20		R3404	MPE/iX Release 5.0 (Major Software Release)[2]
			Threshold Manager (B3474A)
			HPINSTAL/iX (B3159A)
			CD Extensions for MPE/iX (B3683A)
			ALLBASE/REPLICATE (2494B)
			AppleTalk Services for the HP 3000 (J2244A)
			ACT API 4.0 (32077A)
			C2 Security Monitor (B3175A)
			FDDI 3000/iX Network Link (J2245A)
			SNA/Token Ring Link/iX (J2249A)
			NetWare v3.11 for the HP 3000 (32020B)
	X.50.24	R3440	MPE/iX-Express 1 based on Limited Release 5.0
			Fast/Wide SCSI Adapter (2869A)
			Fast/Wide SCSI Disk 2GB x1 (C3550T-R Option 002)
			Fast/Wide SCSI Disk 2GB x2 (C3551T-R Option 002)
			Fast/Wide SCSI Disk 2GB Upgrade Kit (C3554U)

MPE/iX Product Releases (continued)

Release	SUBSYS	Date Code	Product(s) Introduced/Added
C.50.00		R3504	MPE/iX Release 5.0 (Core Software Release)[1] Workload Manager (B3879AA) Shared Globals ARPA Bundling
	C.50.02	R3513	MPE/iX-Express 2 based on General Release 5.0 MOVER Utility Information Access SQL/iX LASERRX/MPE ALLBASE/4GL Developer and Runtime
	C.50.03	R3538	MPE/iX-Express 3 based on General Release 5.0 Information Access Server/iX Symbolic Debug/iX ALLBASE/SQL G.1 IMAGE/SQL G.1 New Systems: HP 3000 Series 969KS/x00 (x = 1 through 4) HP 3000 Series 996/80, 996/x00 (x = 1 through 8) New SCSI Disks: ST31200N (1Gb Single-Ended) ST31230N (1Gb Single-Ended, Low Profile) ST32550N (2Gb Single-Ended) ST15150N (4Gb Single-Ended) ST32550W (2Gb Fast/Wide) ST15150W (4Gb Fast/Wide)

MPE/iX Product Releases (continued)

Release	SUBSYS	Date Code	Product(s) Introduced/Added
C.55.00		R3628	MPE/iX Release 5.5 (Major Software Release)[2] HP Loader Dependent Libraries Subsystem Dump Facility HP Stage/iX HP Patch/iX TurboSTORE/iX 7x24 True-Online Backup HP Optical Disk Libraries: C1150B - 40GB C1160B - 80GB C1170B - 100GB TCP/IP Network Printer Support Telnet/iX Server DTS/TIO Dynamic Configuration Online System Device Configuration

1. As of MPE/iX Release 5.5, the naming convention for Core Software Releases will be known as *Platform Software Releases*.
2. As of MPE/iX Release 5.5, the naming convention for Major Software Releases will be known as *Non-Platform Software Releases*.

Supported Releases

Important Details Please Read

The naming conventions for the different types of releases have been changed slightly to clarify the type of release being discussed. The terms used to describe or refer to the releases are:

Mainline Release	A mainline release involves the recompilization and reintegration of all software release components (FOS, SLT, and SUBSYS tapes). The release number is changed (e.g., 5.0 or 5.5) and the update "UU" field of the V.UU.FF is changed. There are two types of mainline releases: *Platform* and *Non-Platform*.
Platform Release	A platform release (previously also known as a "core" release) is a subset of a mainline release. Typically, the release number ends with a "0" such as 5.0. Platform releases are *automatically distributed* to all customers with support contracts.
Non-Platform Release	A non-platform release (previously known as a "major" release) is a subset of the mainline release. The release number typically ends with a "5" such as 5.5. Non-platform releases must be *explicitly ordered* by customers.

Both platform and non-platform releases can be referred to as *mainline releases* when not discussing distribution or extended support life.

The following matrix provides information on the supported Commercial Systems MPE/iX mainline releases. It lists the currently supported releases and the SPUs they are supported on. The matrix also provides all known factory support termination dates. When a mainline release becomes unsupported, the factory will not provide any support services for that release. Online calls are not accepted and patches are not created: customers are advised to roll to a supported release.

The support life of a release is dependent on its distribution type. Platform releases (such as MPE/iX 4.0 or MPE/iX 5.0 General Release), are automatically distributed to all customers on support contract with HP, and are supported for 24 months, or 12 months after the following Platform release, whichever is longer.

Non-platform releases (such as MPE/iX 5.5), are available upon request to all customers on support contract with HP, and are supported for 12 months, or 6 months after the following mainline release, whichever is longer.

Supported System Release Matrix

Supported Releases	Supported Systems	Support Termination Date
@Release 4.0 (B.40.xx)	920,922,925,930,932,935,948,949,950,955,960, 980-100,980-200,980-300,980-400,9x7,9x7LX, 9x7SX/RX,990, 990DX,992/100,992/200,992/300, 992/400, 992/100DX,992/200DX,992/300DX, 992/400DX	February 1, 1997
@Release 5.0 (C.50.xx)	920,922,925,930,932,935,948,949,950,955,960, 980-100,980-200,980-300,980-400,9x7,9x7LX, 9x7RX,9x7SX,9x8LX,9x8RX,959KS/100,959KS/200, 959KS/300,959KS/400,987/150RX, 987/150SX, 987/200RX,987/200SX,990,990DX, 992/100, 992/200,992/300,992/400,992/100DX, 992/200DX, 992/300DX,992/400DX,991DX, 995/100DX, 995/200DX,995/300DX,995/400DX, 995/500DX, 995/600DX,995/700DX,995/800DX, 991CX, 995/100CX,995/200CX,995/300CX, 995/400CX, 995/500CX,995/600CX,995/700CX, 995/800CX	March 1, 1997*
Release 5.5 (C.55.xx)	920,922,925,930,932,935,939,948,949,950,955, 960,9x7,9x7LX,9x7RX,9x7SX,9x8LX,9x8RX, 959KS/100,959KS/200,959KS/300,959KS/400, 969KS/100,969KS/200,969KS/300,969KS/400, 969KS/120,969KS/220,969KS/320,969KS/420, 980-100,980-200,980-300,980-400,987/150RX, 987/150SX,987/200RX,987/200SX,990,990DX, 992/100,992/200,992/300,992/400,992/100DX, 992/200DX,992/300DX,992/400DX,991DX, 995/100DX,995/200DX,995/300DX,995/400DX, 995/500DX,995/600DX,995/700DX,995/800DX, 991CX, 995/100CX,995/200CX,995/300CX, 995/400CX, 995/500CX,995/600CX,995/700CX, 995/800CX 996/80,996/100,996/200,996/300, 996/400,996/500,996/600,996/700,996/800	August 1, 1997*

@ Platform Releases are denoted by an @ sign.
* Minimum support life.

12

Catalog of User Documentation

Introduction

This chapter provides listings of customer manuals for the HP 3000 computer system. The listings are divided into two sections:

- "MPE/iX 5.5 New, Updated, or Obsoleted Manuals," which lists all manuals that have been introduced or changed at the the time of the MPE/iX 5.5 Release.
- "Manual Sets," which lists manuals by manual set in alphabetical order. Some manuals appear in more than one manual set. New or updated manuals for the MPE/iX 5.5 Release are indicated by an asterisk (*).

  For detailed information on a particular manual or manual set, please refer to the *MPE/iX Documentation Guide* (32650-90144).

If your contract includes Material-Based Services, you will receive both software and manual revisions. For additional copies of new or revised manuals, you can order Manual Update Services (MUS).

Many of the learning products listed in this chapter can be individually ordered by calling HP Parts Direct Ordering at 800-227-8164. Specify the customer order number of the manual you are interested in when ordering.

MPE/iX 5.5 New, Updated, or Obsoleted Manuals

This section lists customer manuals introduced, updated, or obsoleted from MPE/iX-Express 2 based on General Release 5.0 through MPE/iX 5.5 Release.

MPE/iX 5.5 New, Updated, or Obsoleted Manuals

Manual Title	Customer Order No.	Latest Edition	Obsoleted
ALLBASE/DB2 CONNECT User's Guide	30700-90001	12/90	Obsoleted
ALLBASE/TurboCONNECT Administrator's Guide	36385-90001	12/90	Obsoleted
Asynchronous Serial Communications Programmer's Reference Manual	32022-61001	7/96	
Communicator 3000 MPE/iX Release 5.5	30216-90124	7/96	
Configuring and Managing Host-Based X.25 Links	36939-61004	7/96	
Configuring and Managing MPE/iX Internet Services	32650-90835	7/96	
Configuring Systems for Terminal, Printer, and Other Serial Devices	32022-61000	7/96	
HP 3000 MPE/iX Installation, Update, and Add-On Manual	36123-90001	4/96	Obsoleted
HP 3000 MPE/iX System Software Maintenance Manual Release 5.5	30216-90223 R3628[1]	7/96	
HP 3000 PowerPatch Installation Manual	30216-90185	06/96	Obsoleted
HP 3000 PowerPatch Reference Manual	30216-90184	9/94	Obsoleted
HP ARPA File Transfer Protocol User's Guide	36957-61002	9/95	
HP Link Editor/iX Technical Addendum	32650-09476	10/95	
HP SNA Products Remote System Configuration Guide	J2220-61025	3/95	
HP Telnet/iX User's Guide	36957-90154	7/96	
MPE/iX Commands Reference Manual *(Volumes I and II)*	32650-60238	7/96	
MPE/iX Documentation Guide	32650-90144	7/96	
MPE/iX Error Messages Manual *(Volume II)*	32650-60237	7/96	
Native Mode Spooler Reference Manual	32650-90166	7/96	
NetWare 3.11 for HP 3000 Installing and Administering Guide	32020-90027	7/96	
Performing System Management Tasks	32650-90004	7/96	
STORE and TurboSTORE/iX Products Manual	B5151-90001	7/96	
System Startup, Configuration, and Shutdown Reference Manual	32650-90042	7/96	
Using CD-ROM to Update Your HP 3000 System Software	B3159-90001	1/95	Obsoleted

1) The R*nnnn* number changes with each release. R3628 is for the MPE/iX 5.5 Platform Release.

Manual Sets

This section lists customer manuals by manual set in alphabetical order. Some manuals appear in more than one manual set. New or updated manuals for the MPE/iX 5.5 Release are indicated by an asterisk (*).

Manual Title	Customer Order No.	Latest Edition	Obsoleted
SYSTEM RELEASE COMMUNICATORS			
*Communicator MPE/iX Release 5.5	30216-90124	7/96	
Communicator MPE/iX General Release 5.0	30216-90124	1/95	
Communicator MPE/iX Limited Release 5.0	30216-90124	4/95	
Communicator MPE-Express 3 Based on MPE/iX Limited Release 5.0	30216-90189	11/95	
Communicator MPE-Express 2 Based on MPE/iX Limited Release 5.0	30216-90189	4/95	
Communicator MPE-Express 1 Based on MPE/iX Limited Release 5.0	30216-90189	9/94	
Communicator MPE-Express 4 Based on MPE/iX Release 4.0	30216-90186	6/94	
Communicator MPE-Express 3 Based on MPE/iX Release 4.0	30216-10639	3/94	
Communicator MPE-Express 1 Based on MPE/iX Release 4.0	B3484-90003	9/93	
Communicator MPE/iX Release 4.5	30216-90123	11/92	
Communicator MPE/iX Release 4.0	30216-90104	6/92	
SYSTEM MANAGEMENT CORE MANUAL SET (36367A)			
CI Programming Quick Reference Pocket Card	32650-90269	12/90	
*Configuring and Managing MPE/iX Internet Services	32650-90835	7/96	
*Configuring Systems for Terminals, Printers, and Other Serial Devices	32022-61000	7/96	
EDIT/3000 Reference Manual	03000-90012	8/80	
HP 3000/iX Network Planning and Configuration Guide	36922-61023	4/94	
HP TELNET/iX Client User's Guide *Replaced by the HP Telnet/iX User's Guide (36957-90154)*	36957-90152	12/94	Obsoleted
*HP Telnet/iX User's Guide	36957-90154	7/96	
Installing and Managing HP ARPA File Transfer Protocol Network Manager's Guide	36957-61001	6/92	
KSAM/3000 Reference Manual	30000-90079	I 8/86	
Manager's Guide to MPE/iX Security	32650-90474	4/94	
Managing Spooler Operations Quick Reference Pocket Card	32650-90268	4/94	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
*MPE/iX Commands Reference Manual *(Volumes I and II)*	32650-60238	7/96	
*MPE/iX Documentation Guide	32650-90144	7/96	
*MPE/iX Error Messages Manual *(Volumes I, II, and III)*	32650-60237	7/96	
MPE/iX Quick Reference Guide	32650-90032	6/92	
*Native Mode Spooler Reference Manual	32650-90166	7/96	
New Features of MPE/iX: Using the Hierarchical File System	32650-90351	4/94	
NS3000/iX Operations and Maintenance Reference Manual	36922-61005	4/94	
*Performing System Management Tasks	32650-90004	7/96	
Performing System Operation Tasks	32650-90137	4/94	
QUERY/V Reference Manual	30000-90042	5/87	
*STORE and TurboSTORE/iX Products Manual *(For MPE/iX 5.5 Release)*	B5151-90001	7/96	
TurboIMAGE/XL Database Management System Reference Manual	30391-90001	4/94	
Up and Running with ALLBASE/SQL	36389-90011	6/92	
Using KSAM/XL	32650-90168	4/94	
SYSTEM MANAGEMENT CORE PLUS MANUAL SET (36368A)			
Customizing Terminal and Printer Type Files with the Workstation Configurator	5959-2870	2/94	
FCOPY Reference Manual	32212-90003	6/92	
MPE/iX Glossary of Terms and Acronyms	32650-90146	6/92	
MPE/iX Shell and Utilities User's Guide	36431-90002	10/92	
MPE/iX System Utilities Reference Manual	36250-90081	4/94	
SORT-MERGE/XL General User's Guide	32650-90082	11/87 (U 7/88)	
System Startup, Configuration, and Shutdown Reference Manual	32650-90042	4/94	
Troubleshooting Terminal, Printer, and Other Serial Device Connections	32022-61002	10/93	
User's Guide to MPE/iX Security	32650-90472	4/94	
Using the Node Management Services (NMS) Utilities	32022-90041	4/94	
Volume Management Reference Manual	32650-90045	4/94	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
SYSTEM DOCUMENTATION FOR THE HP 3000 SERIES 9X8LX (B3813AA)			
HP 3000 Series 9X8LX Computer Systems: Commands Reference	B3813-90011	4/94	
HP 3000 Series 9X8LX Computer Systems: Getting Started	B3813-90003	4/94	
HP 3000 Series 9X8LX Computer Systems: Task Reference	B3813-90009	4/94	
HP 3000 Series 9X8LX Computer Systems: Understanding Your System	B3813-90001	4/94	
MPE/iX Day to Day Tasks/9X7LX Pocket Card	A1707-90004	6/92	
SYSTEM DOCUMENTATION FOR THE HP 3000 SERIES 9X7LX (B3483A)			
HP 3000 & HP 9000 PA-RISC Customer Computer Support Log	A1703-90012	9/91	
MPE/iX Day to Day Tasks/9X7LX Pocket Card	A1707-90004	6/92	
Preparing Additional Software Products for Use on the HP 3000 Series 9X7LX/Software Installation Guide	36123-90014	6/92	
Setting Up and Maintaining Your System	A1707-90001	6/92	
Understanding Your System/9X7LX	A1707-90003	6/92	
Using Your System	A1707-92002	6/92	
SYSTEM DOCUMENTATION FOR THE HP 3000 SERIES 9X7 and 9X8			
HP 3000 & HP 9000 PA-RISC Customer Computer Support Log	A1703-90012	9/91	
HP 3000 9X7 Installation and Configuration Guide	Non-purchasable product	9/91	
HP 3000 Series 9X7 PA-RISC Computer Systems/Operator Handbook	A1707-90010	12/92	
HP 3000 Series 9X8LX/RX Computer Systems Installation and Configuration Guide	A2051-90006	10/93	
MPE/iX HP 3000 Series 9X7 Software Startup Manual	36123-90015	6/92	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
ALLBASE/SQL CORE MANUAL SET (36372A)			
ALLBASE NET User's Guide	36216-90031	4/94	
ALLBASE/SQL Advanced Application Programming Guide	36216-90100	4/94	
ALLBASE/SQL C Application Programming Guide	36216-90023	6/92	
ALLBASE/SQL COBOL Application Programming Guide	36216-90006	6/92	
ALLBASE/SQL Database Administration Guide	36216-90005	4/94	
ALLBASE/SQL FORTRAN Application Programming Guide	36216-90030	6/92	
ALLBASE/SQL Message Manual	36216-90009	4/94	
ALLBASE/SQL Pascal Application Programming Guide	36216-90007	10/89	
ALLBASE/SQL Performance and Monitoring Guidelines	36216-90102	4/94	
ALLBASE/SQL Reference Manual	36216-90001	4/94	
HP PC API User's Guide for ALLBASE/SQL and IMAGE/SQL	36216-90104	4/94	
ISQL Reference Manual for ALLBASE/SQL and IMAGE/SQL	36216-90096	4/94	
Up and Running with ALLBASE/SQL	36389-90011	6/92	
GENERAL USAGE CORE MANUAL SET (36373A)			
*MPE/iX Commands Reference Manual *(Volumes I and II)*	32650-60238	7/96	
MPE/iX Quick Reference Guide	32650-90032	6/92	
Using the 900 Series HP 3000: Advanced Skills	32650-60126	6/92	
Using the 900 Series HP 3000: Fundamental Skills	32650-60125	6/92	
HARDWARE MANUALS			
HP 3000 & HP 9000 PA-RISC Computer Systems/System Support Log	09740-90013	5/90	
HP 3000 CS 99x/890/T500 Families Operator's Guide	A1809-90009	11/93	
HP 3000 Series 920 Computer System Operator Handbook	A1702-90003	6/90	
HP 3000 Series 922 and Series 932 Family Computer Systems Operator Handbook	A1027-90019	12/90	
HP 3000 Series 9X7 PA-RISC Computer Systems/Operator Handbook	A1707-90010	12/92	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
HP OPEN DESKMANAGER MANUALS (B3606A)			
HP OpenDesk is now sold separately from FOS.			
LANGUAGES MANUALS			
Addendum to the HP RPG/iX Reference Manual	30318-90016	9/95	
HP Business BASIC/XL Migration Guide	32715-60002	10/89	
HP Business BASIC/XL Reference Manual	32715-60001	10/89	
HP C Programmer's Guide	92434-90002	8/92	
HP C/iX Library Reference Manual	30026-90001	10/92	
HP C/iX Reference Manual	31506-90005	6/92	
HP COBOL II/XL Programmer's Guide	31500-90002	7/91	
HP COBOL II/XL Quick Reference Guide	31500-90003	7/91	
HP COBOL II/XL Reference Manual	31500-90001	7/91	
HP FORTRAN 77/iX Programmer's Guide	31501-90011	6/92	
HP FORTRAN 77/iX Reference	31501-90010	6/92	
HP Pascal/iX Programmer's Guide	31502-90002	6/92	
HP Pascal/iX Reference Manual	31502-90001	6/92	
HP RPG/iX Pocket Guide	30318-90002	10/89	
HP RPG/iX Programmer's Guide	30318-60001	7/89	
HP RPG/iX Reference Manual	30318-60002	12/93	
HP RPG/iX Utilities Reference Manual	30318-60003	10/89	
HP Symbolic Debugger/iX User's Guide	31508-90003	6/92	
MPE/iX Developer's Kit Reference Manual *(Volumes 1 and 2)*	36430-60001	4/94	
MPE/iX Shell and Utilities Reference Manual *(Volumes 1 and 2)*	36431-60001	4/94	
MPE/iX Shell and Utilities User's Guide	36431-90002	10/92	
The POSIX.1 Standard: A Programmer's Guide *by Fred Zlotnick*	ISBN-0-8053-9605-5		
MIGRATION CORE MANUAL SET (30231A)			
HP COBOL II/XL Migration Guide	31502-60004	10/88	
HP FORTRAN 77/iX Migration Guide	31502-60004	6/92	
HP Pascal/XL Migration Guide	31502-60004	11/87	
Introduction to MPE XL for MPE V Programmers	30367-60004	10/89	
Introduction to MPE XL for MPE V System Administrators	30367-90003	12/90	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
Migration Process Guide	30367-90007	4/90	
MPE V to MPE XL: Getting Started Mentor's Guide	30367-60002	10/89	
MPE V to MPE XL: Getting Started Self-Paced Training	30367-60002	10/89	
SPL to HP C/XL Migration Guide	30231-60001	10/89	
Switch Programming Guide	32650-60030	11/87	
MPE/iX DEVELOPER'S KIT MANUAL SET (36530A)			
The MPE/iX Developer's Kit has been bundled with the core operating system (FOS). All associated manuals are now in the Languages Manual Set.			
NETWORKING MANUALS			
APPC Subsystem on MPE/XL Node Manager's Guide	30294-61002	6/92	
Asian SNA IMF/XL User Support Guide (Japanese)	30293-60221	4/90	
Asian SNA IMF/XL User Support Guide (Korean)	30293-60231	4/90	
Asian SNA IMF/XL User Support Guide (Taiwanese)	30293-60211	4/90	
Configuring and Managing Host-Based X.25 Links	36939-61004	7/96	
*HP ARPA File Transfer Protocol User's Guide	36957-61002	11/95	
HP OpenView System Manager Manager's Guide	36936-61002	12/90	
HP OpenView System Manager User's Guide	36936-61001	12/90	
*HP SNA Products Remote System Configuration Guide	J2220-61025	3/95	
HP SNA Server/Access User's Guide	30254-61000	12/87	
HP SNADS/XL HP Desk Gateway Administrator's Guide	32006-61001	8/90	
HP SNADS/XL HP Desk User Support Guide	32006-61004	8/90	
HP SNADS/XL Node Manager's Guide	32006-61002	8/90	
HP SNADS/XL SNA/XL Distribution Services: Electronic Mail User Support Guide	32006-61003	8/90	
HP SNMP/XL User's Guide	36922-61029	3/94	
LU 6.2 API Application Programmer's Reference Manual	30294-61000	6/92	
Managing Host-Based X.25 Links Quick Reference Card	36939-61003	9/94	
NetIPC 3000/XL Programmer's Reference Manual	36920-61005	10/89	
NetWare 3.11 for the HP 3000 Installing and Administering Guide	32020-90027	7/96	
NetWare Concepts (for NetWare V.3.11)	J2240-61005	3/93	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
NetWare for the HP 3000 Concepts (for NetWare V.3.01B)	32020-61005	4/91	
NetWare for the HP 3000 Installation (for NetWare V.3.01B)	32020-61001	4/91	
NetWare for the HP 3000 System Administration (for NetWare V.3.01B)	32020-61002	4/91	
NetWare for the HP 3000 System Administration MPE/iX Supplement (for NetWare V.3.01B)	32020-61015	6/92	
NetWare for the HP 3000 System Error Messages (for NetWare V.3.01B)	32020-61006	4/91	
NetWare for the HP 3000 User Basics for DOS Workstations (for NetWare V.3.01B)	32020-61003	4/91	
NetWare for the HP 3000 Utilities Reference (for NetWare V.3.01B)	32020-61004	4/91	
NetWare Installation Manual (for NetWare V.3.11)	J2240-61001	3/93	
NetWare Printer Server (for NetWare V.3.11)	J2240-61008	3/93	
NetWare Troubleshooting and System Messages (for NetWare V.3.11)	J2240-61006	3/93	
NetWare User Basics for DOS Workstations (for NetWare V.3.11)	J2240-61003	3/93	
NetWare Utilities (for NetWare V.3.11)	J2240-61004	3/93	
NS Cross-System NFT Reference Manual	36920-61003	1/89	
NS3000/iX Error Messages Reference Manual	36923-61000	4/94	
NS3000/iX NMMGR Screens Reference Manual	36922-61003	4/94	
RJE User/Programmer Reference Manual	30295-61001	8/90	
RJE/XL Node Manager's Guide	30295-61002	8/90	
SNA DHCF/XL Application Programmer's Manual	36935-61003	11/89	
SNA DHCF/XL Diagnostic Messages Manual	36935-61004	11/89	
SNA DHCF/XL Node Manager's Guide	36935-61002	11/89	
SNA DHCF/XL User Support Guide	36935-61001	11/89	
SNA IMF Programmer's Reference Manual	30293-61005	6/92	
SNA IMF/XL Node Manager's Guide	30293-61000	6/92	
SNA Link/iX Node Manager's Guide	30291-61000	3/94	
SNA NRJE Node Manager's Guide	30292-61000	10/92	
SNA NRJE User/Programmer Reference Manual	30292-61001	10/92	
Using HP OpenView DTC Manager	D2355-90001	12/90	
Using NS3000/iX Network Services	36920-90008	5/94	
Using SNA IMF Pass Thru	30293-61008	12/90	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
PROGRAMMER PRODUCTIVITY TOOLS MANUALS			
BRW Reference Manual	35360-90051	1/92	
BRW Tutorial	35360-90201		
Getting Started with HP Software Revision Controller (SRC)	30234-60002	11/88	
Getting Started with TRANSACT	32247-60002	7/88	
HP ALLBASE/4GL Developer Administration Manual	30601-64201	5/92	
HP ALLBASE/4GL Developer Quick Reference Manual	30601-90210	5/92	
HP ALLBASE/4GL Developer Reference Manual (Volume 1)	30601-90202	5/92	
HP ALLBASE/4GL Developer Reference Manual (Volume 2)	30601-90204	5/92	
HP ALLBASE/4GL Developer Self-Paced Training Guide	30601-90203	5/92	
HP ALLBASE/4GL Run-Time Administration Manual	30602-64201	5/92	
HP ALLBASE/4GL Software Update Notice (B.06)	5961-7797	2/93	
HP ALLBASE/Query User's Guide	92534-64001	10/89	
HP Browse/XL User's Guide	36384-60001	10/90	
HP EDIT Quick Reference Guide	30316-90005	12/90	
HP EDIT Reference Manual	30316-90001	12/90	
HP Search/XL User's Guide	36383-60001	10/90	
HP Software Revision Controller (SRC) Implementation Guide	30234-60002	11/88	
HP Software Revision Controller (SRC) Quick Reference Card	30234-60002	11/88	
HP Software Revision Controller (SRC) User's Guide	30234-60002	11/88	
HP Toolset/XL Reference Manual	36044-60001	1/84	
HP TRANSACT Reference Manual	32247-60003	4/94	
INFORM/V User's Guide	32246-60002	3/88	
Learning HP EDIT	30316-90002	12/90	
REPORT/V User's Guide	32245-60001	2/85	
Virtuoso COBOL Sample Library Reference Manual	30426-60001	5/88	
Virtuoso Code Generator Reference Manual	30422-60001	5/88 (U 10/89)	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
PROGRAMMING CORE MANUAL SET (36369A)			
Berkeley Sockets/iX Reference Guide	32650-90372	2/94	
Compiler Library/XL Reference Manual	32650-60014	10/88	
Getting Started as an MPE/iX Programmer	32650-90008	6/92	
HP Link Editor/XL Reference Manual	32650-90030	12/90	
*HP Link Editor/iX Technical Addendum	32650-09476	10/95	
MPE Segmenter Reference Manual	32650-60026	I 8/86	
MPE/iX Shell and Utilities Reference Manual *(Volumes 1 and 2)*	36431-60001	4/94	
PROGRAMMING CORE PLUS MANUAL SET (36370A)			
Accessing Files Programmer's Guide	32650-90017	6/92	
Asynchronous Serial Communications Programmer's Reference Manual	32022-61001	7/96	
Command Interpreter Access and Variables Programmer's Guide	32650-90011	4/94	
Data Entry and Forms Management System VPLUS Reference Manual	32209-60002	U 11/87	
Data Types Conversion Programmer's Guide	32650-60010	10/89	
High-Level Screen Management Intrinsics Library (Hi-Li) Reference Manual	32424-60001	11/87	
HP DTC Technical Reference Manual	5961-9820	8/93	
Interprocess Communication Programmer's Guide	32650-60011	11/87	
KSAM/3000 Reference Manual	30000-90079	I 8/86	
Message Catalogs Programmer's Guide	32650-60012	11/87	
*MPE/iX Error Messages Reference Manual *(Volumes I, II, and III)*	32650-60237	7/96	
MPE/iX Intrinsics Reference Manual	32650-90028	4/94	
Native Language Programmer's Guide	32650-90022	4/90	
Process Management Programmer's Guide	32650-60011	11/87	
QUERY/V Reference Manual	30000-90042	5/87	
Resource Management Programmer's Guide	32650-60011	11/87	
Trap Handling Programmer's Guide	32650-60010	12/88	
TurboIMAGE/XL Database Management System Reference Manual	30391-90001	4/94	
User Logging Programmer's Guide	32650-60012	11/87 (U 7/88)	
Using KSAM XL	32650-90168	6/92	
Using VPLUS/V: An Introduction to Forms Design	32209-60002	8/86	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
STANDALONE MANUALS			
ALLBASE/DB2 CONNECT User's Guide	30700-90001	12/90	Obsoleted
ALLBASE/TurboCONNECT Administrator's Guide *Replaced by HP IMAGE/SQL Administration Guide (36385-90001)*	36385-90001	12/90	Obsoleted
AutoRestart/XL User's Guide	36375-90001	10/92	
Dictionary/3000 Reference Manual	32244-61000	12/84	
HP Easytime/XL Quick Reference	B1940-90601	6/92	
HP Easytime/XL User's Guide	B1940-90002	9/91	
Getting Started with HP IMAGE/SQL	36385-90008	12/94	
HP CD Extensions for MPE/iX Installation and Administration Guide	B3683-90001	4/94	
HP IMAGE/SQL Administration Guide	36385-90001	12/94	
HP Motif/iX Programmer's Supplement	36394-90001	4/94	
HP Motif/iX System Administrator's Supplement	36394-90002	4/94	
HP Security Monitor/iX Manager's Guide	32650-90455	4/94	
HP Security Monitor/iX User's Guide	32650-90454	4/94	
Magneto-Optical Media Manager User's Guide	36398-90001	4/94	
Mirrored Disk/iX User's Guide	30349-90003	6/92	
Silhouette Reference Manual	30302-60003	12/88	
SORT-MERGE/XL Programmer's Guide	32650-90080	10/89	
SPU Switchover/XL User's Guide	36378-90001	12/90	
STORE and TurboSTORE/iX Manual *(For MPE/iX 5.0 Release)*	30319-90001	4/94	
*STORE and TurboSTORE/iX Products Manual *(For MPE/iX 5.5 Release)*	B5151-90001	7/96	
System Debug Reference Manual	32650-90013	10/89	
The HP 3000: Logging On and Off	32650-90098	11/87	
TurboIMAGE/V Database Management System Reference Manual	32215-90050	12/85	
TurboIMAGE/XL Database Management System DBChange Plus Technical Addendum for MPE/iX Release 4.0	36386-90005	6/92	
TurboIMAGE/XL Database Management System DBChange Plus User's Guide	36386-90001	12/90	
Using the HP 3000 Workload Manager	B3879-90001	12/94	
VPLUS/Windows Programmer's Guide	36393-90002	12/90	

Manual Title	Customer Order No.	Latest Edition	Obsoleted
SYSTEM DICTIONARY MANUALS			
Data Dictionary Managing Information Networks Primer	5958-8527	11/86	
HP System Dictionary/XL COBOL Definition Extractor Reference Manual	32257-90001	12/87	
HP System Dictionary/XL General Reference Manual (*Volume 1*)	32256-90004	12/87 (U 5/88)	
HP System Dictionary/XL General Reference Manual (*Volume 2*)	32256-90005	12/87 (U 5/88)	
HP System Dictionary/XL Intrinsics Reference Manual	32256-90003	12/87 (U 5/88)	
HP System Dictionary/XL SDMAIN Reference Manual	32256-90001	12/87 (U 5/88)	
HP System Dictionary/XL Self-Paced Customer Training	32254-91001	U 8/87	
HP System Dictionary/XL Utilities Reference Manual	32256-90003	12/87 (U 5/88)	
SYSTEM RELEASE MANUALS			
*Communicator 3000 *(Updated for each release of the HP 3000 system software.)*	30216-90124	7/96	
HP 3000 MPE/iX Installation, Update, and Add-On Manual *(Replaced by the HP 3000 MPE/iX System Software Maintenance Manual Release 5.5 (30216-90223R3628)*	36123-90001	4/94	Obsoleted
*HP 3000 MPE/iX System Software Maintenance Manual Release 5.5	30216-90223 R3628[1]	7/96	
HP 3000 PowerPatch Installation Manual *(Replaced by the HP 3000 MPE/iX System Software Maintenance Manual Release 5.5 (30216-90223R3628)*	30216-90185	11/95	Obsoleted
HP 3000 PowerPatch Reference Manual *(Replaced by the HP 3000 MPE/iX System Software Maintenance Manual Release 5.5 (30216-90223R3628)*	30216-90184	9/94	Obsoleted
Using CD-ROM to Update Your HP 3000 System Software *(Replaced by the HP 3000 MPE/iX System Software Maintenance Manual Release 5.5 (30216-90223R3628)*	B3159-90001	1/95	Obsoleted

1) The R*nnnn* number changes with each release. R3628 is for the MPE/iX 5.5 Platform Release.

Index