HP 3000 Commercial Systems

FIELD SUPPORT INTERNALS TRAINING

XL RELEASE 3.0 (Major Release B.30.00)

HEWLETT PACKARD
19111 PRUNERIDGE AVENUE, CUPERTINO, CA 95014

Part No. 30216-90003
* HP Internal Use Only *

Printed in U.S.A. 1/91
R3102

CONTENTS

SECTION 1
XM TRUSTED USER INTERFACE

COMPATIBILITY ISSUES 1-1
MACROS 1-3
DATA TYPES 1-3

SECTION 2
850 LOGONS/600 VT SESSIONS

OVERVIEW 2-1
HISTORY 2-1
OPERATING SYSTEM INTERNAL CHANGES FOR 850 LOGONS 2-2
SPECIAL CONSIDERATIONS FOR 600 VT SESSIONS 2-7

SECTION 3
SECURE SYSTEM

PRODUCT OVERVIEW 3-1
DISCRETIONARY ACCESS CONTROL 3-2
LOGON ACCESS SECURITY 3-28
STREAMJOB 3-32
SECURITY AUDITING 3-35

SECTION 4
MULTIPROCESSOR SYSTEM

INTRODUCTION 4-1
HARDWARE OVERVIEW 4-1
SOFTWARE OVERVIEW 4-2

SECTION 5
SPU SWITCHOVER/XL

HARDWARE REQUIREMENTS 5-1
INSTALLING AND CONFIGURING SPU SWITCHOVER/XL 5-2
DISK LOCKING 5-2
SPU COMMUNICATION 5-3
HANDLING FAILURES 5-3
DATA STRUCTURES AND MACROS 5-4

SECTION 6
DTC/TIO ENHANCEMENTS

PRODUCT ENHANCEMENTS 6-1

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

PREFACE

This document contains internals training articles for XL Release 3.0 (Major Release B.30.00). The software is supported on the series 922, 925, 930, 932, 935, 949, 950, 955, 960, and 980-100, 980-200.

XM TRUSTED USER INTERFACE

SECTION 1

The transaction management (XM) subsystem has been enhanced on MPE XL release 3.0 to support default XM user log files and to include trusted user interfaces to XM from several subsystems. This interface consists of a group of XM procedures that may be accessed by several trusted users including: TurboIMAGE/XL, KSAM/XL, CM KSAM, User Logging, and the Serial Write Queue.

The XM system log files have shrunk in size and now only contain system data structures, such as directory nodes, file label tables, and disk allocation maps. The addition of the default XM user log file has increased the number of XM log files from three to four. However, the four log files occupy the same amount of disk space as the three log files used previously. Nonsystem files (TurboIMAGE/XL, KSAM/XL, CM KSAM, User Logging, and Serial Write Queue) are attached to the XM user log files and protected by user transaction facilities. Files are attached to the default XM user log file for their home volume set.

System transactions are typically very short in duration. On the other hand, user transactions may consist of multiple TurboIMAGE/XL intrinsics that together define a logical database transaction. One advantage provided by the trusted user interface is that programmers can now define their own logical database transactions.

Three new intrinsics have been added to TurboIMAGE/XL to handle logical database transactions: DBXBEGIN, DBXEND, and DBXUNDO. The TurboIMAGE/XL intrinsics inside a DBXBEGIN/DBXEND block are considered as one logical database transaction by the transaction manager. If something undesirable occurs while executing the intrinsics in this block, the programmer can test for this within the code and dynamically rollback the entire logical database transaction using the DBXUNDO intrinsic. Dynamic rollback provides more timely recovery of databases to a logically consistent state than was previously possible through the DBRECOV utility.

COMPATIBILITY ISSUES

The transaction manager keeps four log files per volume set: system, residual 1, residual 2, and user. All XM log files reside on the master volume of a volume set. The log file format for XM log files on releases prior to 3.0 is different from the format used on 3.0.

Forward Migration

When a customer updates to 3.0, the XM log files on the system volume set are automatically converted to the new format during the system installation. The XM log files residing on each nonsystem volume set are converted to the 3.0 format the first time they are opened after the update.

To avoid difficulties, the customer should perform a control-a shutdown before updating the operating system to 3.0. The shutdown will empty the XM residual logs. The forward migration code will not migrate any volumes until these logs are empty. When they are empty, all currently mounted volumes will be migrated together. Additional volumes will be migrated individually when they mount.

If the customer forgets to perform a control-a shutdown before the update, when the updated system is booted the residual logs may contain information about volumes not yet mounted. Since the forward migration code waits until all volumes with log records to be recovered are mounted, the customer should

make sure that all volumes in each volume set he or she wants to migrate are mounted so that the forward migration can occur.

The forward migration code does the following tasks:

- Shrinks the size of the system XM log file and the two residual XM log files.
- Creates a user XM log file.
- Updates the file labels of all nonsystem XM files in the volume set so that changes made to these files will be logged to the XM user log file instead of the XM system log file.

Status messages are displayed on the console to indicate the progress of the migration. When all changes have been made permanent, the message "XM Log File Migration Completed Successfully" is displayed on the console. If an error or system abort occurs before this message is displayed, all changes will be rolled back.

Backwards Migration

If the customer wants to migrate from 3.0 back to a prior release, before shutting down 3.0, he or she should run the XM Log File Backwards Migration Utility (XMMIGBAK.PUB.SYS) on all volume sets to convert the log files back to the pre-3.0 format. This utility should first be run on all the nonsystem volume sets and then be run on the system volume set immediately prior to shutting down the system for the backdate. The other time this utility should be used is before moving a nonsystem volume set to a system running a pre-3.0 release.

The backwards migration utility reverses the disk format changes made during forward migration. If a volume set containing 3.0-format XM log files is mounted on a 2.x version of MPE XL, when the master volume mounts the message "XM User Log Not Compatible With O/S" will appear on the console. The volumes in the volume set will mount in the unknown state and the files on the volumes will not be accessible. If a volume set containing 3.0-format XM log files is mounted on a 1.x version of MPE XL, it will mount successfully since the 1.x releases do not contain the code to recognize this incompatibility. However, you should be aware that nonsystem XM files on the volume set cannot be accessed without risk of corrupting the files.

Backwards migration must be explicitly invoked by a system administrator who has SM or OP capability. The utility is executed with the following command:

```
:RUN XMMIGBAK.PUB.SYS;INFO="<volume_set_name>"
```

where <volume_set_name> is the name of the nonsystem volume set to be migrated.

Several restrictions apply to the use of XMMIGBAK:

- The volume set must be open and all volumes in the set must be mounted.
- There can be no activity on the volume set. All files, including directory files, must be closed.

XMMIGBAK reverses the work done by the forward migration code by enlarging the system and residual log files, purging the user log file, and updating the file labels of all nonsystem XM files on the volume set so

that changes made to these files will be logged in the XM system log, not the XM user log. When it is finished doing these tasks, it closes the volume set so it cannot be accidentally accessed on the 3.0 system.

Changes made during the backwards migration are not permanent until the message "XM Log File Migration Completed Successfully" appears on the $STDLIST device. If an error (for example, all volumes in the volume set are not mounted, or one or more files in the volume set are currently open) or system abort occurs before this message is displayed, all changes will be rolled back.

MACROS

A collection of DAT macros to analyze XM log files was introduced with release 2.1 and described in the document "DAT Macros for XM System Log Analysis". These macros have been modified to give information about the XM user log file as well as the XM system log file.

For example, you can use the XM_SET_ENV macro to set up an XM macro environment for analyzing the XM system or XM user log file. If you do not provide any parameters to the macro, the macro will ask you to select a volume set and also to select which XM log file (1=system, 3=user) to examine. These fields may also be specified as parameters to the macro call.

DATA TYPES

The data type for XM system log files is contained in the type declaration "XMSYSLOGLAYOUT"; the data type for XM user log files is described in the type declaration "XM_CIRCULAR_USER_LOG_LAYOUT". XM user log files consist of a header and two log halves. The header is one page in size and contains information about the log version, operating system creation version, log half locations, checkpoint numbers, and so on. The size of the log half depends on the disk type; this data structure solely contains log records.

850 LOGONS/600 VT SESSIONS

SECTION 2

OVERVIEW

This article will focus on the system structures which have changed in order to support 850 active logons of which 600 logons can be virtual terminal (VT) connections. It will focus on the resource constraints which have increased to support greater connectivity. It will also discuss some of the resource constraints which will need to increase for further connectivity improvements.

In order to increase logon connectivity for release 3.0, specific resource constraints have been removed. Below is a table which compares some existing limitations for MPE XL Release 2.1 and 3.0.

Increased Limits	2.1	3.0
Max # of processes (PCB entries)	1559	3119
Max # of configured terminals(nmmgr)	600	850
Max # of IDD/ODD entries	741	1254
Max # of inbound PDX entries	360	611

Existing Limits	2.1	3.0
Max # of ldevs (sysgen)	999	999
Max # of dymamic devices (sysgen)	999	999
Max # of RINS	5459	5459
Max # of DST entries	16384	16384
Max # of JMAT entries	858	858
Max # of TCP connections	1024	1024
Max # of NS server processes	700	700
Max # of outbound PDX entries	360	360

NMMGR configuration limits have been set to be consistent with the 850 logon maximum. The total number of configured terminals (nailed and non-nailed) cannot exceed 850.

A data structure called the PDX (Path Descriptor Extension) is required for each node to which an XL system can connect. In order to support a maximum of 600 virtual terminal connections from a maximum of 600 unique nodes, the maximum number of inbound PDX entries must also be increased to (at least) 600. For release 3.0, the PDX is expanded to its maximum -- 611 entries. With this increase in PDX entries, a maximum of 600 VT connections from 600 different remote nodes is possible. The most common use of this feature being the support of up to 600 VT connections on an XL host from up to 600 different PCs.

For each incoming VT connection, a dynamic device is allocated on the HP 3000/XL system. Although the value of the number of dynamic devices is set to a default of 332, it is user configurable (through SYSGEN) on release 3.0 to a maximum of 999.

The VT enhancements available with release 3.0 have specifically improved VT performance. VT "Fast Path" is a rewrite to the VT software such that VT now uses the Native Mode File System. This change to VT has significantly reduced the VT path length and CPU utilization.

There are a few other parameter limitations that have not changed with release 3.0, but are worth mentioning. NS 3000/XL still supports a maximum of 1024 TCP connections and with a maximum of 700 total NS server processes.

Two tables which increased in order to support 850 logons are the Process Control Block (PCB) and the Input/Ouput Device Directory (IDD/ODD). The changes made to support these increases are described throughout this article.

The resource constraint on 3.0 which inhibits increasing the logon connections to an amount significantly greater than 850 logons is the Job Master Table (JMAT). There is one entry for each job or session running on the system. The maximum number of entries this table currently has is 858.

HISTORY

The following sections provide background information on previous limits and changes required to support increased connectivity.

OS Issues

Release 2.1 supports 600 logons with 50% active. A logon uses two processes, the JSMAIN and the command interpreter (CI). However, a VT logon uses three processes, the JSMAIN, the CI and the VTSERVER process. It is estimated approximately 300 of these logons can run a single child process before process limits are reached. The calculation follows : (600 * 2) + 300 = 1500 processes. The 1500 process count does not include system processes used by the operating system. The number of operating system processes required varies based on configuration and product installation. A conservative estimate is to reserve 100 processes for operating system usage.

On release 2.1, the table which restricts the maximum number of concurrent processes is the Process Control Block (PCB). This table is restricted by the size of the compatibility mode (CM) data segment. The maximum size of a CM data segment (DST) is 32767 CM words. A CM word is 16-bits or 2 bytes. Each PCB entry is 21 CM words. The PCB header is 21 CM words. The maximum number of PCB entries available for release 2.1 are (32767 - 21) / 21 = 1559.

NS Issues

Various performance improvements to the NS services software and OS have been made which can improve performance for supporting 600 VT sessions:

- Native Mode IOWAIT (first available release 2.1)
- Native Mode NetIPC (first available release 2.1)
- VT "Fast Path" (first available release 3.0)

Both the rewrite of IOWAIT and NETIPC software into Native Mode available with Release 2.1, have helped to improve the overall performance of the NS Services by shortening the NS path length from NetIPC down through the NS transport and by reducing switching between Compatibility Mode and Native Mode. The availability of NETIPC and IOWAIT in Native Mode has specifically helped the NS VT service as it is also in Native Mode and no longer must switch to Compatibility Mode for its NETIPC and IOWAIT calls. The VT enhancements available with release 3.0 have specifically improved VT performance. VT "Fast Path" change has significantly reduced the VT path length and CPU utilization.

OPERATING SYSTEM INTERNAL CHANGES FOR 850 LOGONS

This section describes various operating system tables that were expanded to support increased connectivity.

PCB Expansion

As previously mentioned, an increase in the PCB table size is needed to to support greater than 1559 processes. On Release 3.0 the PCB DST is now 65535 CM words. This increases the number of PCB entries to: (65535 - 21) / 21 = 3119. Other DSTs still observe the 32767 CM word limit. Several changes were required to support the PCB DST expansion. In order to access all the entries in the 64K word PCB DST the emulator and translator changed to interpret the X register as unsigned for the load and store system table (LST/SST) instructions. All 16 bits are used to generate up to a 65535 CM word offset. The list below describes the valid and invalid access methods for the PCB table.

```
Valid PCB Access Methods
========================
1. Load/Store System Table (LST/SST)

Invalid PCB Access Methods (will cause a system abort on Release 3.0)
=====================================================================
1. DST Relative Move (MFDS, MTDS, MDS)
2. Extended Addressing (LSEA, SSEA, SDEA, LDEA)
3. Split Stack/DB Relative Addressing - the EXCHANGEDB procedure was used
   to point DB to an alternate DST
```

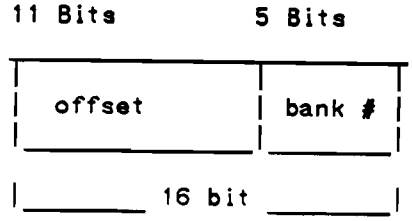
The DST relative move instructions and EXCHANGEDB method of accessing the PCB table require information in the DST table. The DST table describes all the DSTs including the PCB DST. The PCB DST entry #3 in the DST table has been zeroed out. It was decided that it is better to not allow these methods to work than have them work part of the time. Letting these access methods work for only half of the PCB table would be harder to diagnose as a problem. When these invalid access methods are used they will result in a system abort (CM SUDDENDEATH 16 DST VIOLATION). Since CM does not have a method for distinguishing the cause of a DST violation it is not possible to avoid a system failure. These failures should not occur through operating system access to the PCB table. There may be some customers who have written privilege mode programs or third-parties who access the PCB directly. They should be encouraged to use the Architected Interface Facility (PN 36374A).

Release 3.0 supports 850 active logons. An active logon uses three processes, the JSMAIN, the CI and the child process. However, an active VT logon uses four processes, the JSMAIN, the CI, the VTSERVER and the child process. With 600 active VT logons, it is estimated that 2400 processes are used. With a maximum of 3119 processes, it is not quite possible to achieve 850 active logons if 600 of these active logons are VT logons: 3119 - 100 processes system overhead - (600 * 4) - (250 * 3) = -131 processes. With 600 active VT logons it is estimated that a maximum of 206 additional active logons is possible.

CM Bank 0 Changes

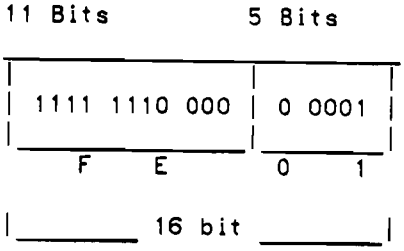
The PCB table is too large to fit into CM Bank 0. CM Bank 0 is also referred to as the CM absolute object (KSO #190). It is contained in a native mode object. A memory bank on the "Classic" 3000 is 64K 16-bit words. Additional logical banks are therefore needed in the CM absolute object. DSTs are not allowed to

cross bank boundaries. The PCB DST is now placed in Bank 1 of CM absolute object and its 64K CM word size will fill the entire bank. Previously there was only CM Bank 0 on MPE XL. The CM System Global Area has a pointer in word 3 which points to the base of the PCB table. This pointer is in "11/5" format. In the 16-bit word the most significant 11 bits represent the offset and the least significant 5 bits represent the bank number.



The emulator supports the "11/5" pointer format by converting it into a 32 bit offset. This offset is added to the virtual address of CM Bank 0 to get the virtual address of the PCB table. Below is an example of the process the emulator follows to convert the "11/5" pointer address to a virtual address for the PCB table.

1. CM System Global word 3 has a value of $FE01



2. Take the bank number in the lower 5 bits and place it in a temporary register. Zero fill the lower 5 bits overwritting the bank #.

```
$0000 FE00
```

3. Add an octal %1000 to the 16 bit offset. %1000 is the difference in CM words between CM Bank 0 base and the CM System Global Table.

```
$0000 FE00
+    %1000
-----------
$0001 0000
```

4. Replace the upper 16 bits with the bank number right justified and zero filled.

```
$XXXX 0000
 0001
----------
$0001 0000 CM word offset
```

5. Multiply this value by 2 to generate a byte offset. Add the byte offset to the base address of CM Bank 0 (KSO #190). This will generate the virtual address of the PCB table.

```
    $00010000
    x       2
    ---------
    $00020000
+ $b.80000000
-------------
  $b.80020000 virtual address of PCB table
```

The "11/5" pointer format has been in use by CM for some time now. Previously there was only CM bank 0 on MPE XL. Below is an additional example of the SIR table "11/5" pointer conversion.

1 & 2. The SIR table is in CM BANK 0. The system global entry has a value of:

```
$0000 AA00  "11/5" pointer (bank # = 0)
```

3. Add %1000

```
$0000 AA00
+    %1000
----------
$0000 AC00
```

4. Replace upper 16 bits with bank #.

```
$XXXX AC00
$0000
----------
$0000 AC00
```

5. Multiply by 2 for byte offset. Add to virtual address of CM Bank 0.

```
    $0000AC00
    x       2
    ---------
    $00015800
+ $b.80000000
-------------
  $b.80015800 virtual address of SIR table
```

The above examples were generated from a prerelease version of MPE XL release 3.0. These values may change in the future. The formula for the conversion will still apply.

Mapping PCB Offsets to Pin Numbers

There are many places in the operating system (CM modules) where a PCB offset (PCBPT) is converted to a PIN number by dividing the "PCBPT" by the PCB entry size. Since the PCB DST has been expanded the "PCBPT" must be declared as a LOGICAL. In SPL the expression (PCBPT/%25) will generate different code depending on how "PCBPT" is declared. If it is declared as an INTEGER, then SPL will generate a DIVI instruction. The DIVI instruction will interpret "PCBPT" as a signed number. PCB entries in the second half of the expanded "DST" will cause the division to yield a negative pin number (which is incorrect). If "PCBPT" is declared as a LOGICAL, SPL will generate an LDIV instruction, and the division will yield the correct result in all cases. The same is true for the SPL expression (PIN * %25) will cause and

overflow when "PCBPT" is declared as an INTEGER. Declaring "PCBPT" as a logical will result in the correct offset into the PCB table.

Once again, the CM operating system code which was converting the PCB offset to a pin number and vice versa has been changed to return the correct results.

CONVERT_DST Changes

CONVERT_DST is used in native mode to take a DST number and an offset (in CM words) and return the virtual address of that offset. This procedure was changed to exception the PCB DST. It no longer accesses the PCB virtual address from the DST table (which is now nil). The virtual address of the PCB (KSO #66) is retrieved from the KSO table. Pointers to data structures can reside in the KSO table even though they may not necessarily be a separate object.

IDD/ODD Entry Size Is Reduced

The Input Device Directory (IDD) and Output Device Directory (ODD) are data structures which hold information about the owners of each non-sharable device on the system. The IDD structure follows: a header of 12 CM words, a header entry of four words for each configured device which allows input, and a subentry of 40 words for each input device which has been FOPENed. The ODD is identical except it manages output devices. Because terminals support both input and output, they are managed in both the IDD and ODD. Therefore, the IDD and ODD place a limit on the number of terminals which can be opened concurrently on MPE XL Release 2.1 to: (32640-12)/(40+4) = 741.

In order to support 850 logons on MPE XL Release 3.0 the number of IDD/ODD entries had to be increased. This was accomplished by decreasing the size of the IDD/ODD subentry. Since the release of the Native Mode Spooler in MPE XL 2.1 the IDD/ODD table is no longer used to manage spoolfiles. Specific fields within an IDD/ODD subentry which were used for managing spoolfiles could be removed to reduce the subentry size. Of the 40 CM words in a subentry, 18 have been removed. By eliminating these fields, the number of devices which can be supported becomes: (32640-12)/(22+4) = 1254. In order to support additional entries in the future the entry may be further restructured. The current entry format is listed below.

```
XDD_SUBENTRY_TYPE = CRUNCHED RECORD

    XDDS_WORD0          : XDDS_WORD0_TYPE ($0.0 @ 2.0);
    JOB_TYPE            : BIT2 ($2.0 @ 0.2);
    JOB_NUMBER          : BIT14 ($2.2 @ 1.6);
    USER_NAME           : XDDS_NAME_TYPE ($4.0 @ 8.0);
    ACCOUNT_NAME        : XDDS_NAME_TYPE ($c.0 @ 8.0);
    JOB_NAME            : XDDS_NAME_TYPE ($14.0 @ 8.0);
    FILE_NAME           : XDDS_NAME_TYPE ($1c.0 @ 8.0);
    DFID_OUT            : BOOLEAN ($24.0 @ 0.1);
    DFID_NUMBER         : BIT15 ($24.1 @ 1.7);
    ODDS_FORMS_IN_FILE  : BOOLEAN ($26.0 @ 0.1);
    IDDS_DATA           : BOOLEAN ($26.1 @ 0.1);
    XDDS_FILLER2        : BOOLEAN ($26.2 @ 0.1);
    HEAD_INDEX          : BIT13 ($26.3 @ 1.5);
    DEVICE              : BIT16 ($28.0 @ 2.0);
    NEXT_SUBENTRY       : BIT16 ($2a.0 @ 2.0);
    END ($0.0 @ 2c.0)
```

Impact on Debugging Techniques

Some of the changes required to expand the PCB table had an impact on the methods of accessing PCB information in DAT/DEBUG.

```
The following methods will NOT work when accessing the PCB.
===========================================================
1. DD dst.offset
2. DA offset              ** will not work beyond CM BANK 0 **
3. DSTVA (dst.offset)
4. Fixed low memory cell 4 no longer contains current pcb pointer.
   This has been moved to processor specific data areas for MP.

The following methods will work when accessing the PCB.
=======================================================
1. PCB (pin)              ** returns the virtual address of a PCB entry **
2. KSO_POINTER(#66)       ** returns the virtual address of PCB table **
```

Impact on CM Privilege Mode Users and VABS

Outside of HP there are a number of privileged third party software packages that may reference the PCB and other CM tables directly. All of these software products need to be examined and possibly modified. These privilege mode applications should investigate using the Architected Interfaces (product number 36374A) to access information in the PCB and other CM tables. This will allow these applications to continue to operate independent of any future changes to the CM PCB and other tables.

SPECIAL CONSIDERATIONS FOR 600 VT SESSIONS

This section highlights various configuration guidelines and network management concerns.

Network Shutdown

The ability to abort 600 VT sessions simultaneously is limited by both the NS Services and NS Transport software because of the large number of processes, port procedures, server processes, data structures, etc. that must be released. Attempting to destroy all these processes and structures severely strains internal resources such that the simultaneous shutdown may not succeed and can, under certain circumstances, result in a network or system hang. Because of the level of complexity of resolving this issue through software changes, a solution was not possible in Release 3.0. Customers are strongly recommended to avoid shutting down the network when there are many active connections (or VT sessions).

Specific Configuration Changes

With release 3.0, certain default configuration values are not specifically set to support 600 VT connections. The following changes must be made:

- Increase the Maximum Number of Dynamic Devices (MAXDYNIO) in SYSGEN. The default value is 332. Configure one dynamic device for each incoming VT session and one dynamic device for each LAN or Point-to-Point link.

- Increase the Maximum Number of VTSERVER processes with the NSCONTROL SERVER= command. Execute this command after each start of the NS Services on the system. The default value for VTSERVER processes is 300.

VT Connections from PCs

When an HP 3000/XL system supports a large number of VT connections, it is generally because most of the VT traffic to the HP 3000/XL system is from PCs. There are three special considerations to be aware of when planning such a network. These are:

- PC Window Size
- PC Retransmission Policy
- PC Connection Assurance Timer

There are two PC products that contain the Virtual Terminal service: OfficeShare and NS Services 2.0 for DOS. NS Services 2.0 for DOS is part of the LAN Manager product family.

PC Window Size

Because of limitiations within the PC, such as memory size, the PC VT service must be sensitive to the window size it uses. OfficeShare negotiates a window size of 1. In general, this causes an immediate acknowledgement for each packet. This results in high network traffic. With NS Services 2.0 for DOS, the PC window size is increased to 3 to reduce network traffic while not overburdening the limited resources of the PC.

PC Retransmission Policy

When there is significant network traffic, PCs with an aggressive retransmission policy can start retransmitting at a quick pace further aggrevating the network congestion problem. All of the OfficeShare products have a non-configurable aggressive retransmission policy. With the LAN Manager Family of products, including NS Services 2.0 for DOS, the PC retransmission policy has a more aggressive backoff policy to better accomodate congested networks.

PC Connection Assurance Timer

PCs periodically send a Connection Assurance (CA) packet when the connection is idle to determine whether the connection to the HP 3000 system is still up. All of the OfficeShare products have a CA timer value of 60 seconds. When there is significant network traffic and many PCs have idle connections to the HP 3000/XL system, a low CA timer value can aggrevate an already congested network. The LAN Manager Family of products will provide a configurable CA timer in the next set of releases. NS Services 2.0 for DOS is scheduled to re-release in late 1990. Then the CA timer value will default to 10 minutes. Ten minutes is the recommended CA timer value for the PC when used with the HP 3000/XL system.

In order to support more than 300 VT connections on the HP 3000/XL system, PCs must be updated to one of the products from the LAN Manager Family. Please consult the product data sheets in the current HP AdvanceNet Specification Guide for more information on these products.

SECURE SYSTEM

SECTION 3

PRODUCT OVERVIEW

Much of this information is available in the FOS manual set, but it is spread across seven (7) different manuals. It is provided here in an attempt to centralize it for your reading enjoyment.

The changes in MPE XL provided within Secure System address three primary areas:

- Discretionary Access Control
- Logon Access Security
- Security Auditing

Discretionary Access Control

File access in MPE XL can be made more secure by the implementation of a Discretionary Access Control (DAC) mechanism. This mechanism is discretionary, in that it is up to the owner of the object to grant access rights to whomever he wishes and access is based solely on this, not on the enforcement of some mandatory rules (e.g. a file access matrix).

The C2 classification of the Trusted System Evaluation Criteria requires that a trusted system "shall define and control access between named users and named objects", and the enforcement mechanism "shall be capable of including or excluding access to the granularity of a single user". The DAC guideline from the National Computer Security Center recommends the use of Access Control Lists (ACLs) as the best way to meet the DAC requirements. We will have Access Control Definitions as our DAC mechanism on MPE XL. Access Control Definitions (ACDs) are extensions of Access Control Lists in the sense that an ACD not only contains a list of users who can access an object (e.g. a file or a device), but it may also contain restrictions such as the day and time at which a user can access the object. Each protected object can have an attached ACD which specifies who can or cannot access the object. At the time the object is initially accessed (e.g. file open), this function is evaluated to determine if the access attempt should be granted or denied.

Only a subset of the possible features of Access Control Definitions will be implemented in MPE XL on release 3.0. This implementation is fully compatible with the MPE V/E V-Delta-4 implementation of ACDs. On first release, files and devices are the only objects protected by ACDs, and an ACD will only contain a list of users and the access modes each user has to the file or device.

Logon Access Security

Three new features have been added to improve logon access security.

- A new CI command for manipulating user passwords.
- Enhancements to password prompting.
- A STREAMJOB privilege-mode interface.

Security Auditing

The following security auditing enhancements will be implemented on the system logging facility:

- logging of password changes
- logging of system logging configuration
- logging of restore
- logging of printer access failure
- logging of ACD changes
- logging of stream initiation
- logging of user logging
- logging of process initiation
- auditability by named user

With these enhancements to auditing, system log files will be filling up faster. This will require more attention to archive old log files.

DISCRETIONARY ACCESS CONTROL

File access in MPE XL can be made more secure by the implementation of a Discretionary Access Control (DAC) mechanism. This mechanism is discretionary, in that it is up to the owner of the object to grant access rights to whomever he wishes and access is based solely on this, not on the enforcement of some mandatory rules (e.g. a file access matrix).

The C2 classification of the Trusted System Evaluation Criteria requires that a trusted system "shall define and control access between named users and named objects", and the enforcement mechanism "shall be capable of including or excluding access to the granularity of a single user". The DAC guideline from the National Computer Security Center recommends the use of Access Control Lists (ACLs) as the best way to meet the DAC requirements. We will have Access Control Definitions as our DAC mechanism on MPE XL. Access Control Definitions (ACDs) are extensions of Access Control Lists in the sense that an ACD not only contains a list of users who can access an object (e.g. a file or a device), but it may also contain restrictions such as the day and time at which a user can access the object. Each protected object can have an attached ACD which specifies who can or cannot access the object. At the time the object is initially accessed (e.g. file open), this function is evaluated to determine if the access attempt should be granted or denied.

Only a subset of the possible features of Access Control Definitions will be implemented in MPE XL on release 3.0. This implementation is fully compatible with the MPE V/E V-Delta-4 implementation of ACDs. On first release, files and devices are the only objects protected by ACDs, and an ACD will only contain a list of users and the access modes each user has to the file or device.

ACD Definition

Two definitions of an ACD follow. The first definition is a general definition in which an ACD is a function that defines who has access to an object, when a user has access to the object, and what modes of access the user has for that object. Also, it defines what types of permission a user has to manipulate the ACD associated with the object. This definition is the definition that will apply to ACDs in the future. Note that while in first release, only files and devices have ACDs; in a later release, other objects such as directory nodes, user ports, etc., may also be protected by ACDs.

The second definition is a subset of the first one. An ACD is defined as a list of users and the access modes each user has to the file/device. It includes a list of users who can copy and read the ACD itself. It does not include restrictions on when the user can access the file/device. The general definition is provided for completeness.

GENERAL DEFINITION. An ACD is defined as follows:

```
ACD              = (pair_spec)

pair_spec        =  pair_spec
                    pair_spec ; pair_spec
                    modes: subject_spec

subject_spec     =  subject
                    ( subject_spec )
                    subject_spec set_operator subject_spec
                    subject_spec boolean_operator restrictions

subject          =  user_name
                    user_name.grouping_name
                    program_name
                    process_name


set_operator     =  To be defined.


boolean_operator =  To be defined.


restrictions     =  To be defined.


modes            =  modes
                    modes , modes
                    access
                    permission

access           =  R | W | A | L | X | None

permission       =  RACD (read the ACD)
```

The access and permission types may be expanded in the future, especially when ACDs are used to protect objects other than files and devices. Also, not all accesses are applicable to all object types.

SPECIFIC DEFINITION. The ACD includes a list of users and the types of access each user or group of users has to the file/device. It also includes all those users who can copy or read the ACD itself. The ACD is defined as pairs of user specifications and access modes allowed to those users.

ACD = (*modes* : *user specifications* ; *modes* : *user specifications*; . . .)

An ACD is defined as follows:

```
ACD                  = (pair_spec)

pair_spec            =  pair_spec
                        pair_spec ; pair_spec
                        modes: user_specifications

user_specifications  =  user
                        user_specifications set_operator user_specifications

user                 =  user_name.account_name
                        @.account_name
                        @.@

set_operator         =  , (which is being used as U)

modes                =  modes
                        modes , modes
                        access
                        permission

access               =  R | W | A | L | X | None

permission           =  RACD (read the ACD)
```

Each pair consists of *modes* : *user specification*. *Modes* can be a combination of any of the following types of access and permissions, separated by "," :

- *R* : READ access.
- *W* : WRITE access.
- *L* : LOCK access.
- *A* : APPEND access.
- *X* : EXECUTE access.
- *NONE* : no access at all.
- *RACD* : COPY or READ the ACD permission.

EXAMPLES:

R,W,L
A,R
NONE
X,R

User specifications are defined as:
1. User: defined by

- *username.accountname*

That is, a fully qualified user name.

EXAMPLES:

RACHAEL.MSE

DANNY.CSSO

2. User Grouping: two forms of user grouping are allowed. The "@" wildcard character is used to represent these two groupings:

- *@.accountname* : which represents all users who are associated with the account defined by *accountname*.
- *@.@*: which represents all users in the system.

EXAMPLES:

@.PAYROLL : includes all users of the PAYROLL account.

@.@ : includes all users in the system.

3. Users and user groups separated by "," : which defines the union of all users specified. There can be any combination of users as defined in 1. and 2. separated by ",".

EXAMPLES:

JOHN.DOE, @.PAYROLL

JOE.SMITH, @.PERSONEL, @.@
JOHN.DOE, @.@

Therefore, an ACD can be defined as follows:

- ACD = (R : FARK.DOE ; W,A,L : @.DOE, @.PAYROLL ; R : @.@)
- ACD = (NONE : RAY.DOE, @.CSSO ; R,W : @.@)

Each user specification (e.g., RAY.DOE, or @.@) and associated modes make up an entry in the ACD. For MPE XL first release, the number of entries allowed in an ACD is forty (40). This is twice the maximum in MPE V/E.

FILE ACDs: The OWNER(s) of an ACD associated with a file is:

1. The CREATOR of the file to which the ACD is associated.

2. User with AM capability (in the account where the file resides).

3. User with SM capability.

DEVICE ACDs: The OWNER(s) of an ACD associated with a device is:

1. The System Manager.

Having access to a file or device is different from having permission to access the ACD for that file or device. For example, if a user has read (R) access to a file or device, he/she may or may not have permission to read the ACD for that file, depending on if he/she is granted RACD permission.

The OWNER of an ACD has all permissions to the ACD. That is, the OWNER is the ONLY one who can create, delete, modify, list, and copy the ACD. However, the OWNER can authorize other users to access his/her ACD. By giving users "RACD" permission, the OWNER allows users to READ and COPY the ACD.

RACD is the only permission type allowed to be given to other users in the MPE XL Secure Systems (at first Release).

Where are ACDs stored

An ACD for a file is stored in the file label extension for that file. An ACD for a device or class is stored in the file label extension of the appropriate file in the 3000DEVS account.

How are ACDs Created and Maintained

File/device ACDs can be created and maintained through the :ALTSEC command and through the intrinsic HPACDPUT.

In addition, ACDs can also be created as the result of :COPY, :FCOPY, and :RESTORE when new files are created using these commands.

File ACDs become a part of the permanent file objects and therefore will survive a re-boot or optionally can be STORE'd or FCOPYed to a tape. Device ACDs, however, are not part of permanent objects and must be re-applied each time the system is re-booted. It is suggested that this be done by adding the appropriate :ALTSEC commands to create the device ACDs in the SYSSTART file or to a command file which is included in the SYSSTART file.

Maintaining ACDs via Commands

```
                                  [;NEWACD = {(pair_spec)}                    ]
                                             {^acdfilename}                   ]

                                                                [,FILENAME*]]
                                  [;COPYACD = {sourceobjectname} [,LDEV     ]]
                                                                [,DEVNAME   ]]

                      [,FILENAME*]                {(pair_spec)}               ]
ALTSEC  objectname    [,LDEV     ]  [;ADDPAIR = {^acdfilename}                  ]
                      [,DEVCLASS]
                      [,DEVNAME  ]  [;REPPAIR = {(pair_spec)}                 ]
                                                {^acdfilename}                ]

                                    [;DELPAIR = {(userspecification)}         ]
                                                {^acdfilename}                ]

                                    [;DELETE                                  ]
```

* - Default entry

Examples:
```
ALTSEC AFILE;NEWACD=(R,W:OPERATOR.SYS;X:@.@)
ALTSEC 22,LDEV;COPYACD=21,LDEV
```

Maintaining ACDs via Intrinsics

```
                I32       IV      *    O-V
HPACDPUT (status [,itemnum1,item1]
                 [,itemnum2,item2]);
```

Itemnum1 and item1 specify the target file/device ACD to manipulate.

itemnum1	item1
1	Target file name. Character array.
2	Generic file name. Character array.
3	Target UFID. UFID type, a 20-byte record structure. (Note: UFID type is defined in DGLOBFS.FSGLOB.OFFICIAL)
4	Target file number. 16-bit short integer.
5	All devices. Item value not required.
6	Target device. 16-bit short integer.
7	Device name. Character array.
8	Device class. Character array.

Itemnum2 and Item2 specify the input data to be used.

itemnum2	item2
20	Create a new ACD from specification. Character array.
21	Add ACD pairs from specification. Character array.
22	Replace ACD pairs from specification. Character array.
23	Delete ACD pairs from specification. Character array.
24	Delete the ACD. Item value not required.
25	Create new ACD from file. Character array.
26	Copy ACD from file. Character array.
27	Copy ACD from UFID. UFID type.
28	Copy ACD from file number. 16-bit integer.
29	Copy ACD from device number. 16-bit integer.
30	Copy ACD from device name. Character array.
31	Add pairs from file. Character array.
32	Replace pairs from file. Character array.
33	Delete pairs from file. Character array.

How are ACDs Used

ACDs are used to determine if a user trying to access a file/device is authorized to do so. When there is an ACD associated with a file/device, the old access matrix (including lockwords) is not used to determine who can access a file/device. Instead, the ACD is the only mechanism used to determine who can access the file/device.

Thus, ACDs are checked in two different ways: when accessing a file, and when acquiring a device.

Accessing a file

In order for a user to access a file, the system will execute the following checks at HPFOPEN/FOPEN time:

1. The system will check whether the user is any of the following:

   - System Manager (has SM capability),
   - Account Manager (has AM capability), or
   - CREATOR of the file.

If the user is any of these three, then he/she may access the file. Otherwise, the following check is executed by the system.

2. The system checks if there is an ACD associated with the file. If there is one, the system evaluates the ACD to determine whether the user is granted access to the file. This evaluation is done by matching the user to any of the user specifications given in the ACD.

Matching the user to the ACD is done as follows:

- the user name is compared to all the specific names (username.accountname) in the ACD. If the name does not match, then
- the user name is compared to all the account groupings (@.accountname) in the ACD. If not matching account entry is found, then
- the user name is compared to any system grouping present (@.@). If no system entry (@.@) is found in the ACD, then
- the user is not granted access.

If a match is found, the user is given the access and permissions specified in the match entry.

3. If no applicable ACD is found, then the old access matrix and lockword are used to determine if the user is granted access to the file. When a user is denied access to a file, whether because of ACD verification or according to the old access matrix, the same file system error, security violation, is returned to the user.

Special Case :

PRIVILEGED FILES:

1. The system checks whether the process has PM capability, whether the file code matches, and the privilege level is the correct one. If neither one of these conditions hold, then the user is denied access. If all of these conditions hold, then

2. The system checks if there is an ACD associated with the file. If there is one, the system evaluates the ACD to determine whether the user has access to the file.

3. If there is no ACD, then the old access matrix and lockwords are used to determine if the user is granted access to the file.

Acquiring a device

For a user to acquire a device, the system will execute the following checks at HPFOPEN/FOPEN time:

1. The system will check whether the user is System Manager or the process has PM capability, in which case the user is allowed to acquire the device.

2. Otherwise, the system checks if the device has an ACD associated with it. If there is an ACD, then the system evaluates the ACD the same way that is described above. That is, the system tries to match the user's name with any of the user specifications in the ACD. If there is a match, then the user is allowed to acquire the device as specified in the ACD. Otherwise, the user is disallowed acquisition of the device.

3. If there is no ACD, then acquiring the device is done as is done today.

A user FOPENs a device by giving either a device class name, a device name, or a device number.

If the user FOPENs a DEVICE CLASS then, the system will take the first available device which belongs to that device class and perform the checks described above to determine if the user is allowed to acquire the device. If the user is disallowed, then the user will have to try again.

If the user FOPENs a DEVICE NUMBER then, the system will only try to match the user with a user specification in the ACD for the specific device. If there is a match, then the user is granted access. If there is no match then the user is denied access.

When a user is denied access to a device, whether because of ACD verification or according to the old access cheking method, the same file system error, security violation, is returned to the user.

Reporting ACDs via Commands

:LISTF [fileset], 4

The output of this command will show if the file has an associated ACD, and what access the user has according to that ACD.

```
:LISTF FILENAME, 4
*********************
FILE: FILENAME.GROUP.CSSO

SYSTEM     READ:     ANY
SECURITY--WRITE:     AC
 (ACCT)    APPEND:   AC
           LOCK:     AC
           EXECUTE:  ANY

SYSTEM     READ:     GU
SECURITY--WRITE:     GU
 (GROUP)   APPEND:   GU
           LOCK:     GU
           EXECUTE:  GU
           SAVE:     GU

SECURITY--READ:      ANY                    FCODE: 0
 (FILE)    WRITE:    ANY                    CREATOR: **
           APPEND:   ANY                    LOCKWORD: **
           LOCK:     ANY                  **SECURITY IS ON
           EXECUTE:  ANY                  **ACD EXISTS

FOR JOE.CSSO: READ,WRITE,APPEND,LOCK,EXECUTE
```

The "**ACD EXISTS" display is new. Other possible displays in this place are: NO ACD and ACD CORRUPTED.

:LISTF [fileset], -2

This new "-2" option in the :LISTF command will display the content of the ACD associated with the file. If there is no ACD applicable to the file, the output will show "NO ACD".

```
LISTF FILENAME,-2
```

Output for this command will contain all entries of the ACD as follows:

```
FILE = FILENAME      *************** ACD ENTRIES ***************

                     JOE.DOE         : R
                     @.OSE           : R,W,A,L,X
                     @.@             : X
```

The new :LISTFILE command will also contain an option to list the ACD associated with the file. The output for this option is the same as shown above.

:SHOWDEV will also be modified to display ACDs associated with devices.

```
              [devname      ]
:SHOWDEV      [ldev         ] [;ACD]
              [devclassname]
```

An example output from the :SHOWDEV command with the ACD option included is as follows:

```
:showdev 14
LDEV  AVAIL          OWNERSHIP          VOLID          DEN   ASSOCIATION

 14   SPOOLED        SPOOLER OUT
      ACD ENTRIES:   @.@ : R,W,X
```

Obtaining ACD Info via Intrinsics

```
                I32        IV       *     O-V
HPACDINFO(status [,itemnum1,item1]
                 [,itemnum2,item2]
                 [,itemnum2,item2]
                 [,itemnum2,item2]
                 [,itemnum2,item2]);
```

Itemnum1 and item1 specify the target file/device ACD to get infomation.

itemnum1	item1
1	Target file name. Character array.
3	Target UFID. UFID type.
4	Target file number. 16-bit short integer.
6	Target device. 16-bit short integer.
7	Target device name. Character array.

Itemnum2 and Item2 specify the data to be used or returned.

itemnum2	item2
20	Get version number. 16-bit integer.
21	Get number of entries. 16-bit integer.
22	Get first user. Character array.
30	User name as an input. Character array.
31	Get mode ascii. Character array.
32	Get modes bit. 16-bit integer.
33	Get modes evaluated ascii. Character arry.
34	Get modes evaluated bit. 16-bit integer.
35	Get next user. Character array.

Intrinsic 'HPACDINFO' / Manipulating ACDs

There are functions that a user can use to manipulate ACDs. Functions are provided to create, alter, delete and list an ACD. While a user can manipulate ACDs using commands, programmatic interfaces are also provided to manipulate ACDs.

Programmatic interfaces are intrinsics. The existing intrinsic HPFOPEN will be modified to include creating an ACD. A new intrinsic, HPACDINFO will be added which allows security attributes to be returned. In addition another new intrinsic, HPACDPUT will be added to create, delete, modify, copy and list ACD attributes. Both of these intrinsics are compatible supersets of their MPE V/E ounterparts.

HPACDINFO

The HPACDINFO interface requires a status parameter. Following the status parameter are several keyword/keyvalue pairs. The keyword specifies a desired option. The keyvalue is a reference parameter which is dependent on the keyword for its interpretation.

Please see the MPE XL Intrinsics manual for more complete information.

SYNTAX

```
                I32        IV       *     O-V
HPACDINFO(status [,itemnum1,item1]
                 [,itemnum2,item2]
                 [,itemnum2,item2]
                 [,itemnum2,item2]
                 [,itemnum2,item2]);
```

New and Altered CI Commands

ACDs can also be manipulated using commands. The existing :ALTSEC command has been modified to manipulate ACDs associated with files and devices. It has also been modified to allow a user to modify

ACDs associated with privileged files. The following syntax diagram is used to describe the changes to :ALTSEC:

```
                                    [;NEWACD = {(pair_spec)}                    ]
                                               {^acdfilename}                   ]

                                    [;COPYACD = {sourceobjectname} [,FILENAME*]]
                                                                   [,LDEV     ]]
                                                                   [,DEVNAME  ]]

                                                {(pair_spec)}                   ]
                        [,FILENAME*]  [;ADDPAIR = {^acdfilename}                ]
ALTSEC  objectname      [,LDEV     ]
                        [,DEVCLASS]
                        [,DEVNAME ]   [;REPPAIR = {(pair_spec)}                 ]
                                                  {^acdfilename}                ]

                                      [;DELPAIR = {(userspecification)}         ]
                                                  {^acdfilename}                ]

                                      [;DELETE                                  ]
```

* - Default entry

The :SECURE and :RELEASE commands have been modified to return a warning when there is an ACD associated to the file. These two commands are NOT applicable when a file has an ACD associated with it.

The LISTF,4 commands have been modified to display what accesses a user has according to an ACD when there is an ACD associated with the file.

The output for LISTF,4 will appear as follows:

```
:LISTF FILENAME, 4
**********
FILE: FILENAME.GROUP.OSE

SYSTEM     READ:     ANY
SECURITY--WRITE:     AC
 (ACCT)    APPEND:   AC
           LOCK:     AC
           EXECUTE:  ANY

SYSTEM     READ:     GU
SECURITY--WRITE:     GU
 (GROUP)   APPEND:   GU
           LOCK:     GU
           EXECUTE:  GU
           SAVE:     GU
```

```
SECURITY--READ:     ANY                    FCODE: 0
 (FILE)   WRITE:    ANY                    CREATOR: **
          APPEND:   ANY                    LOCKWORD: **
          LOCK:     ANY                  **SECURITY IS ON
          EXECUTE:  ANY                  **ACD EXISTS

FOR JOE.OSE: READ,WRITE,APPEND,LOCK,EXECUTE
```

:SHOWDEV will also be modified to display ACDs associated with devices.

```
SHOWDEV  [ldev          ] [;ACD]
         [devclassname]
```

Example output from the :SHOWDEV command with the ACD option included is as follows:

```
:showdev 14
LDEV  AVAIL          OWNERSHIP          VOLID          DEN   ASSOCIATION

 14   SPOOLED        SPOOLER OUT
      ACD ENTRIES:   @.@ : R,W,X
```

A new option, LISTF,-2 has been added to display an ACD associated with a file.

```
LISTF  [fileset] [,-2]
```

Output for this command will contain all entries of the ACD as follows:

```
FILE = MEMMSTR       ********** ACD ENTRIES **********

                     JOE.DOE          : R
                     @.OSE            : R,W,A,L,X
                     @.@              : X
```

The COPY command will be modified to always copy the ACD associated with the source file to the target file, if one is present. No syntax change is needed for the :COPY command.

```
COPY [FROM=]sourcefile[[;TO=]targetfile]
```

FCOPY will be modified to include the option to copy an ACD.

```
FCOPY [fcopycommand][;COPYACD]
```

STORE and RESTORE have been altered to accommodate ACDs.

Creating an ACD

To create an ACD for a file, the user MUST be:

- The CREATOR of the file, or
- Account Manager of the account where the file resides (a user with AM capability), or
- System Manager (a user with SM capability).

To create an ACD for a device, the user MUST be:

- System manager (SM capability).

Commands to CREATE an ACD

The user can use the following commands to CREATE an ACD:

```
                        [,FILENAME*]  [            {(pair_spec)}            ]
ALTSEC  objectname      [,LDEV     ]  [;NEWACD =   {^acdfilename}           ]
                        [,DEVCLASS]
                        [,DEVNAME ]

    * - Default entry
```

The above command creates an ACD and associates it with a file or device. The ACD itself can be given explicitly in the command (pair_spec), or it can be found in a file (acdfilename). The file containing the ACD information is a file whose content is an ACD. Note that this does not refer to a file which has an ACD attached to it.

Wild cards will be allowed for the filename so a user can associate an ACD with more than one file at a time. Wild card characters allowed and their meanings are the same as in the :LISTF command.

An ACD can be created for all devices in the system by using the following wild card instead of devicenumber/name/class:

- @ : which represents all devices in the system.

Failure to CREATE an ACD Interactively

An attempt to create an ACD will fail under any of the following circumstances:

- The user trying to create an ACD is NOT authorized to do so because
  - the user is not the CREATOR of the file, or

- the user is not the account manager of the account where the file resides (a user with AM capability), or
- the user is not the System Manager (a user with SM capability).
- A user given as part of the ACD does not exist.
- An incorrect type of access mode is given as part of the ACD.
- Nonexistent file/device given as the object for which the ACD is being created.
- File containing the ACD information does not exist.
- User does not have read access to the file containing the ACD information.
- An ACD already exists for the file/device.
- A syntax error has occurred.
- An internal error occurred when trying to CREATE the ACD.

Programmatic interface to CREATE an ACD

The programmatic interface to create an ACD will have the following characteristics:

- Input: the name or number of the file/device and the ACD (or the file name which contains the ACD information) to be attached to the file/device.
- Output: a status representing whether the ACD was created successfully.
- Action: the interface will check whether the user creating the ACD is allowed to do so. If that is the case, it will then verify that the ACD is a valid one. Once this is validated, the interface will allocate the space for the ACD, link it to the file/device definition, and copy the information in the ACD.

This interface will be part of the existing intrinsic HPFOPEN and of a new intrinsic HPACDPUT.

The existing intrinsic FOPEN cannot be modified to add the option of creating an ACD, due to the restrictions imposed by the implementation of FOPEN, so HPACDPUT must be used.

Failure to CREATE an ACD Programmatically

Creating an ACD will fail under any of the following circumstances:

- The user trying to create an ACD is NOT authorized to do so because
  - the user is not the CREATOR of the file, or
  - the user is not the account manager of the account where the file resides (a user with AM capability), or

  - the user is not the System Manager (a user with SM capability).
- A user given as part of the ACD does not exist.
- An incorrect type of access mode is given as part of the ACD.
- Nonexistent file/device given as object for which the ACD is being created.
- File containing the ACD information does not exist.
- User does not have read access to the file containing the ACD information.
- An ACD already exists for the file/device.
- A syntax error has occurred.
- An internal error occurred when trying to CREATE the ACD.

Failure to create an ACD will result in the failure of the execution of the intrinsics invoked.

Reading an ACD

Reading an ACD is performed when copying an ACD and when listing an ACD. Only the OWNER(s) of an ACD and those users to whom the OWNER(s) gives read ACD permission ("RACD" permission) can read (and therefore copy and list) the ACD. Both of these are explained in the following:

Copying an ACD

A user can copy an ACD from one file/device to another file/device only if he/she is able to READ the ACD (has "RACD" permission or is owner) from the source file/device and is allowed to create an ACD for the destination file/device.

EXAMPLE:

There is a file FILEA.XX.DESIGN whose CREATOR is MGR.DESIGN. FILEA has an ACD defined as:

ACD = (R : JOHN.DOE ; RACD: SAM.DOE ; W,A,L : @.DESIGN, @.PAYROLL ; R : @.@)

The following users are the only ones authorized to COPY the ACD:

- MGR.DESIGN (the CREATOR of the file),
- Users in the DESIGN account with AM capability (since they are OWNERs of the ACD),
- Users in the system with SM capability (since they are also OWNERs of the ACD), or
- SAM.DOE who can only copy it to a valid file (that is a file that he is the CREATOR of, or that resides in the DOE account and he has AM capability, or SAM.DOE has SM capability).

Commands to COPY an ACD

The commands to COPY an ACD from one file/device to another are defined as follows:

```
                    [,FILENAME*]                                          [,FILENAME*]]
ALTSEC objectname   [,LDEV     ]  [;COPYACD = (sourceobjectname)          [,LDEV     ]]
                    [,DEVCLASS]                                           [,DEVNAME  ]]
                    [,DEVNAME ]

    * - Default entry
```

Wild cards will be allowed for the target filename so a user can copy an ACD to more than one file at a time. Wild card characters allowed and their meanings are the same as in the :LISTF command.

An ACD can be copied to all devices in the system by using the following wild card instead of targetdevicenumber/targetdevicename:

- @ : which represents all devices in the system.

The COPY command will be modified to always copy the ACD associated with the source file to the target file, if one is present. No syntax change is needed for the :COPY command.

:COPY from=sourcefile;to=targetfile

The FCOPY subsystem will also be modified to add the option of copying an ACD associated with a file when the file is being copied:

:FCOPY from=sourcefile;to=targetfile[;COPYACD]

Note that the copying the file will fail when the user copying the ACD is not authorized to do so. That is, neither the file nor the ACD will be copied even if the user is authorized to copy the file. Also, copying an ACD to a remote system will fail. This is an enhancement that will be done in a later release. So for now, the user will be able to copy just a file to a remote system without the ACD attached to it. The ACD can be copied to the remote system by including it in the contents of a file, or by using STORE/RESTORE.

And, STORE/RESTORE will also be modified to add the option of storing/restoring an ACD associated to a file when that file is being stored/restored as follows:

STORE

The option to copy an ACD when storing a file will be added.

STORE filename ...;COPYACD

When storing a file with the COPYACD option and the file has an ACD associated with it, the ACD will be evaluated to check if the user is granted access to store the file and the ACD (RACD permission is required). If the user is not granted access to copy the ACD, then storing the file will fail. Otherwise, the ACD will be copied from disk into the tape where the file is being stored. If there is no ACD for the file, the file will be stored and a warning will be displayed.

If wild cards are used to define file names, then the system will try to store each of the files with their associated ACD. If in the process there are files for which storing the ACD fails, an appropriate message will be displayed.

RESTORE

The option to copy an ACD when restoring a file will be added.

RESTORE filename . . .;COPYACD

When restoring a file with the COPYACD option and the file has an ACD associated with it, the ACD will be evaluated to check if the user is granted access to restore the file and the ACD (RACD permission is required). If the user is not granted access to copy the ACD, then restoring the file will fail. Otherwise, the ACD will be copied from tape and attached to the restored file on disk. If there is no ACD stored with the file, the file will be restored and a warning will be displayed.

If wild cards are used to define file names, then the system will try to restore each of the files with their associated ACD. If in the process there are files for which restoring the ACD fails, an appropriate message will be displayed.

For COPY, FCOPY, and STORE/RESTORE the default is NOT to copy the ACD associated with a file. Instead, the user must use the COPYACD option explicitly.

```
:SYSGEN

    SysGen> TAPE [MODE = ACD/NOACD]

ISL> INSTALL ACD/NOACD

ISL> UPDATE ACD/NOACD
```

On UPDATE, NOACD (default) means the existing ACDs on disk will be preserved, instead of replacing them with the ones from tape.

Failure to COPY an ACD Interactively

Copying an ACD will fail when:

- The user trying to copy an ACD is not authorized to do so.
- There is already an ACD associated with the destination file/device.
- There is no ACD associated with the source file/device.
- A nonexistent file/device is given as source or destination file/device.
- The user is not authorized to create an ACD on the destination file/device.
- A syntax error has occurred.
- An internal error occurred when trying to COPY the ACD.

Programmatic Interface to COPY an ACD

The programmatic interfaces to copy an ACD have the following characteristics:

- Input: the name or number of the file/device containing the source ACD and the name or number of the file/device where the ACD is being copied to.
- Output: a status representing whether the ACD was copied successfully.
- Action: the interface will check whether the user copying the ACD is allowed to do so. Then, the interface will allocate the space for the new ACD, link it to the file/device definition, and copy the information from the source ACD to the space allocated.

Failure to COPY an ACD Programmatically

Copying an ACD will fail when:

- The user trying to copy an ACD is not authorized to do so.
- There is already an ACD associated with the destination file/device.
- There is no ACD associated with the source file/device.
- A nonexistent file/device is given as source or destination file/device.
- A syntax error has occurred.
- An internal error occurred when trying to COPY the ACD.

Failing to copy an ACD will result in the failure of the execution of the intrinsic invoked.

Listing an ACD

Only the OWNER(s) of an ACD and those users authorized by the OWNER of the ACD can list the ACD. The OWNER of an ACD authorizes who can read the ACD by including a list of users within the ACD who have RACD permission (that is, read permission to the ACD). Use of wildcards will be allowed and used in the same manner as presently used with the :LISTF command.

EXAMPLE:

if the ACD is associated with file FILEA.XX.DESIGN whose CREATOR is MGR.DESIGN is defined as:

ACD = (R : JOHN.DOE ; RACD : SAM.DOE ; W,A,L : @.DESIGN, @.PAYROLL ; R : @.@)

Then, the following users are the only ones authorized to READ the ACD:

- MGR.DESIGN (the CREATOR of the file),

- Users in the DESIGN account with AM capability (the OWNER(s) of the ACD),
- Users in the system with SM capability (the OWNER(s) of the ACD), and
- SAM.DOE (user with "RACD" permission).

Commands to LIST an ACD

Some existing commands have been modified to return the ACD associated with a file/device, or the accesses a user is granted when there is an ACD associated to a file: :LISTF and :SHOWDEV.

The commands to LIST an ACD are defined as follows:

```
:LISTF    filename,-2
```

which will list the entire ACD for filename.

Those wild cards applicable to LISTF in general are still applicable in this case.

```
:LISTF    filename,4
```

This command will be modified to display only those access modes granted to the issuer of the command for the file specified in filename. Those wild cards applicable to LISTF in general are still applicable in this case.

LISTF,4 will be modified so that it will display the access modes a user is granted. This will be done by updating what is displayed under the title "FOR username.accountname: ..." to reflect what accesses the user is granted when the file has an ACD associated with it. In this case a message *** ACD EXISTS *** will be displayed.

```
:SHOWDEV    devicenumber/deviceclass/devicename;ACD
```

which will list the entire ACD for devicenumber/name/class.

Failure to LIST an ACD Interactively

Listing an ACD will fail when:

- The user trying to list an ACD is not authorized to do so.
- A nonexistent file/device is given as file/device.
- A syntax error has occurred.
- An internal error occurred when trying to LIST the ACD (internal errors will be defined in the Internal Specifications).

Programmatic Interface to LIST an ACD

The programmatic interfaces to list an ACD have the following characteristics:

- Input: the name or number of the file/device and the userspecification (to display his/her access modes).
- Output: a status representing whether the ACD information was displayed successfully, and the specific ACD information requested that is associated withthe file/device.
- Action: the interface will check whether the user listing the ACD is allowed to do so. If that is the case, it will return the specific information requested (i.e. access modes).

A user can obtain ACD information by specifying the options to return ACD information. These options are arranged into three groups:

1. The objectname of the ACD to be listed
2. Information to be returned that does not require specification of a user.
3. Information to be returned that does require specification of a user.

Failure to LIST an ACD Programmatically

Listing an ACD will fail when:

- The user trying to list an ACD is not authorized to do so.
- There is no ACD associated with the file/device.
- A nonexistent file/device is given as file/device.
- A syntax error has occurred.
- An internal error occurred when trying to list the ACD.

Failing to list an ACD will result in the failure of the execution of the intrinsic invoked.

Modifying an ACD

This function will provide the user with the capability of modifying an existing ACD. The following changes can be made:

- Add *modes* : *user specifications* pair.
- Replace the *modes* part of an existing pair.
- Delete a *modes* : *user specifications* pair.

Only those *user specifications* : *modes* pairs that are being added/modified need to be specified. The user specification for replacing and deleting must match exactly the entry in the ACD. The OWNER(s) of an ACD is the ONLY one allowed to modify the ACD.

Commands to MODIFY an ACD

The system will search the ACD using the user specifications provided in the commands as the searching key before modifying the ACD.

The commands to MODIFY an ACD are defined as follows:

ADD

```
                    [,FILENAME*] [           {(pair_spec) }]
ALTSEC objectname   [,LDEV     ] [;ADDPAIR = {^acdfilename}]
                    [,DEVCLASS]
                    [,DEVNAME ]

    * - Default entry
```

which will ADD the modes:userspec pairs included in the pair_spec given as part of the command, if the userspec part does not exist in the ACD. If the userspec part already exists, an error will be returned.

The modifications to be made can be given explicitly (pair_spec) or can be found as the contents of acdfilename.

REPLACE

```
                    [,FILENAME*] [           {(pair_spec) }]
ALTSEC objectname   [,LDEV     ] [;REPPAIR = {^acdfilename}]
                    [,DEVCLASS]
                    [,DEVNAME ]

    * - Default entry
```

It will search the ACD trying to find those user specifications given in the (pair_spec) or in acdfilename. If they are found then, the access modes granted to those users in the ACD are REPLACED by those given in the (pair_spec) or in acdfilename. If not found, then an error will be returned.

The modifications to be made can be given explicitly (pair_spec) or can be found as the contents of acdfilename.

DELETE

```
                    [,FILENAME*] [           {(userspecification) }]
ALTSEC objectname   [,LDEV     ] [;DELPAIR = {^acdfilename}        ]
                    [,DEVCLASS]
                    [,DEVNAME ]
```

```
* - Default entry
```

which will DELETE those pairs associated with the userspec specified in the DELPAIR option or in file acdfilename. This command will fail if the userspec does not exist in the ACD.

The userspec to be deleted can be given explicitly (userspec; . . .) or can be found as the contents of acdfilename.

Wild cards will be allowed for the filename so a user can modify more than one ACD at a time. Wild card characters allowed and their meanings are the same as in the :LISTF command.

The ACDs for all devices in the system can be modified by using the following wild card instead of devicenumber/name/class:

- @ : which represents all devices in the system.

For example, if the following ACD is associated with the file FILEA.XX.DESIGN: ACD = (R : SAM.DOE; W : JOE.DOE; NONE : @.DESIGN; X : @.@) then, the ACD will be modified as follows when the following commands are performed.

1. :ALTSEC FILEA.XX.DESIGN;ADDPAIR=(R : JOE.DESIGN)

ACD = (R : SAM.DOE; W : JOE.DOE; R : JOE.DESIGN; NONE : @.DESIGN; X : @.@)

2. :ALTSEC FILEA.XX.DESIGN;REPPAIR=(W : SAM.DOE)

ACD = (W : SAM.DOE; W : JOE.DOE; R : JOE.DESIGN; NONE : @.DESIGN; X : @.@)

3. :ALTSEC FILEA.XX.DESIGN;DELPAIR=(@.DESIGN)

ACD = (W : SAM.DOE; W : JOE.DOE; R : JOE.DESIGN; X : @.@)

4. :ALTSEC FILEA.XX.DESIGN;DELPAIR=(JOE.DOE)

ACD = (W : SAM.DOE; R : JOE.DESIGN; X : @.@)

5. :ALTSEC FILEA.XX.DESIGN;ADDPAIR=(W : JOE.DOE)

ACD = (W : SAM.DOE; R : JOE.DESIGN; W: JOE.DOE; X : @.@)

Failure to MODIFY an ACD Interactively

Modifying an ACD will fail under any of the following circumstances:

- The user trying to modify an ACD is not the OWNER of the ACD.
- When there is no ACD associated with the filename/devicename.
- When adding a *modes* : *user specifications* pair if:
    - The user specification already exists in the ACD.

    - A user to be added does not exist in the system.
- When replacing the access modes in a *modes* : *user specification* pair if:
    - The user specification does not exist.
- When deleting *user specifications*, there is no pair for that user specification in the ACD.
- The file/device to which the ACD is attached does not exist in the system.
- A syntax error has occurred.
- An internal error occurred when trying to MODIFY the ACD.

Programmatic Interface to MODIFY an ACD

The programmatic interfaces to modify an ACD have the following characteristics:

- Input: the name or number of the file/device and either a buffer containing the changes to be made, or the name of the file whose contents are the changes to be made.
- Output: a status representing whether the ACD was modified successfully.
- Action: the interface will check whether the user modifying the ACD is allowed to do so. If that is the case, it will then verify that the modifications requested are valid ones. Once validated, the interface will modify the ACD as requested.

Failure to MODIFY an ACD Programmatically

Modifying an ACD will fail under any of the following circumstances:

- The user trying to modify an ACD is not the OWNER of the ACD.
- When there is no ACD associated with the filename/devicename.
- When adding a *modes* : *user specifications* pair if:
    - The user specification already exists in the ACD.
    - A user given does not exist in the system.
- When replacing the access modes in a *modes* : *user specification* pair if:
    - The user specification does not exist.
- When deleting *user specifications*, user specifications does not exist.
- The file/device to which the ACD is attached does not exist in the system.

- A syntax error has occurred.
- An internal error occurred when trying to MODIFY the ACD.

Failing to modify an ACD will result in the failure of the execution of the intrinsic invoked.

Deleting an ACD

This function allows the user to delete an ACD associated with a file/device. The OWNER of an ACD is the only one allowed to DELETE an ACD.

However, a special feature will be provided so that System Manager (user with SM capability) can reset all ACDs in the system. This is for the extreme case when ACDs are no longer required in the system.

Commands to DELETE an ACD

The commands to DELETE an ACD are defined as follows:

```
                    [,FILENAME*]
ALTSEC objectname   [,LDEV     ]   [;DELACD ]
                    [,DEVCLASS]
                    [,DEVNAME ]

    * - Default entry
```

which will delete the ACD for file/device.

Wild cards will be allowed for the filename so a user can delete more than one ACD at a time.

Wild card characters allowed and their meanings are the same as in the :LISTF command.

The System Manager can delete all the ACDs for devicenumbers/deviceclasses by using the following wild card instead of devicenumber/name/class:

- @ : which represents all devices in the system. This wild card can only be used by the System Manager.

Failure to DELETE an ACD Interactively

Deleting an ACD will fail when:

- The user trying to delete an ACD is not the OWNER of the ACD.
- The user trying to delete all ACDs on the system is not System Manager (a user with SM capability).

- There is no ACD associated with the file/device.
- The file/device given as containing the ACD to be deleted does not exist.
- A syntax error has occurred.
- An internal error occurred when trying to DELETE the ACD.

Programmatic interface to DELETE an ACD

The programmatic interfaces to delete an ACD have the following characteristics:

- Input: the name of the file/device (or devicenumber/name/class).
- Output: a status representing whether the ACD was deleted successfully.
- Action: the interface will check whether the user deleting the ACD is allowed to do so. If that is the case, the interface will delete the ACD.

This interface will be part of the new intrinsic HPACDPUT.

Deleting an ACD through HPACDPUT will be done by using HPS_OPTION_DELETE_ACD.

Failure to DELETE an ACD Programmatically

Deleting an ACD will fail when:

- The user trying to delete an ACD is not the OWNER of the ACD.
- The user trying to delete all ACDs on the system is not System Manager (a user with SM capability).
- There is no ACD associated with the file/device.
- The file/device given as containing the ACD to be deleted does not exist.
- A syntax error has occurred.
- An internal error occurred when trying to DELETE the ACD.

Failing to delete an ACD will result in the failure of the execution of the commands or intrinsics invoked.

Migration of ACDs

Migration of file ACDs are accomplished via the STORE/RESTORE process. Transporting files with ACD are possible both ways, i.e. from MPE V/E system to MPE XL, and vice versa, with the use of the COPYACD parameter in :STORE and :RESTORE commands.

In MPE XL, both CM and NM STORE will recognize this option, with the CMSTORE being able to write and read tape in the MPE V/E format.

As for device ACDs, since they are tied to system configuration of specific systems, device migration from one system to the next is probably not a desirable feature. No device ACD migration is planned with this release,

LOGON ACCESS SECURITY

This section describes the logon access security FOS features which are being implemented for the MPE XL 3.0 security platform release:

- A new CI command for manipulating user passwords.
- Enhancements to password prompting.
- A STREAMJOB privilege-mode interface.

:Password Command

The :PASSWORD command is a new FOS command which allows all users to change the password associated with their user name. Prior to the introduction of this command only account or system managers could manipulate user passwords.

When a user name is associated with a non-blank password, users are re-authenticated before they are allowed to change the password. This re-authentication protects against a person other than a user ID's owner from walking up to an unattended terminal and using the :PASSWORD command within a logged-on session to change a user password. Re-authentication is performed by prompting users for their user passwords and verifying their responses. If an incorrect re-authentication password is entered, the :PASSWORD command terminates after displaying an error message on the user's terminal and a message on the system console.

New passwords are prompted for twice in order to catch nonrecurring typographic errors. All password responses are converted to upper case upon input. If both new password responses are not the same (ignoring case), the password will not be changed and the command will terminate with an error message. New passwords must satisfy password syntax rules and be different than the old user password. These password syntax rules are the same as the rules enforced for the :NEWUSER and :ALTUSER commands: the password must be no longer than eight characters long, begin with a character, and contain only alphanumeric characters. A message indicating whether a user password has been changed is displayed on the user's terminal before the command terminates.

The CI's HPTIMEOUT variable value is enforced while waiting for user responses to password prompts. If this time interval expires while the :PASSWORD command is waiting for a user to enter a password, the session is terminated just as if the time interval expired while waiting at the CI prompt for input.

Because passwords should not be stored in files where they can be discovered by other users, and non-interactive input must be stored in some type of file, HP has restricted the :PASSWORD command to be executable only in interactive sessions whose $STDIN and $STDOUT have not been redirected.

Password privacy would be seriously undermined if the :PASSWORD command could be invoked from within a job file containing the new password. Programs may execute the :PASSWORD command programmatically as long as the program is executing within a session environment which satisfies the

previously stated restrictions. The :PASSWORD command avoids displaying passwords on full duplex terminals by disabling character echo while prompting for passwords and by not allowing passwords to be secified as command line parameters.

:PASSWORD Dialogue Examples

Successful Password Replacement

```
:PASSWORD
ENTER OLD USER PASSWORD:          (old password entered)
ENTER NEW USER PASSWORD:          (new password entered)
ENTER NEW USER PASSWORD AGAIN:    (new password entered again)
PASSWORD WAS CHANGED SUCCESSFULLY.
:
```

Unsuccessful Re-authentication

```
:PASSWORD
ENTER OLD USER PASSWORD:          (incorrect password entered)
INCORRECT PASSWORD.  (CIERR 2502)
PASSWORD WAS NOT CHANGED.
:
```

Unsuccessful New Password Verification

```
:PASSWORD
ENTER OLD USER PASSWORD:          (old password entered)
ENTER NEW USER PASSWORD:          (new password entered)
ENTER NEW USER PASSWORD AGAIN:          (typographic error)
NEW PASSWORD IS NOT CONSISTENT. (CIERR 2503)
PASSWORD WAS NOT CHANGED.
:
```

Interactions with Other Features

Successful invocations of the :PASSWORD command will generate password change log records if system logging of password changes has been enabled.

Compatibility

The MPE XL :PASSWORD command is upwards compatible with the MPE V/E :PASSWORD command. The MPE XL :PASSWORD command differs from the V/E command in three ways. User passwords are changed only when the new password is different than the old password. Secondly, the MPE XL :PASSWORD command is programmatically executable. Thirdly, the CI's hptimeout value is enforced while waiting for user input.

Enhanced Logon Password Prompting

Logon password prompting is being enhanced in two ways. The first enhancement is that all password prompts will contain the account, user, or group name in addition to the type of password (account, user, or group) which is currently displayed. For example, the logon prompt requesting the user password associated with the MANAGER.SYS user ID will be **ENTER USER (MANAGER) PASSWORD:**. The second enhancement is that the password prompting which currently occurs for the :HELLO and :CHGROUP commands will be extended to the following areas:

- :STARTSESS commands within sessions
- STARTSESS intrinsic calls in program executing within sessions
- :STREAM commands within sessions, prompting will occur for each first level :JOB and :DATA command

Password prompting and verification will continue to follow the existing rules for the :HELLO command. Users will be prompted a maximum of three times for each password. Each failure to supply a correct password will cause a message containing the user's identity and the logical device number of the logon device to be displayed on the system console. If the logical device belongs to a device class which has been associated with a user, the error message will be displayed on the $STDLIST of the associated user rather than on the system console. Failure to supply a correct response after being prompted for the same password three times will cause the error message "INCORRECT PASSWORD. (CIERR 1441)" to be displayed on the user's terminal. :STREAM commands will ignore input after this error occurs until either another :JOB or :DATA command is read or an end of file occurs; other commands will terminate after displaying this error message.

Because the :DATA command does not include a group specification, group password prompting does not apply to :DATA commands. Password prompting does not occur when $STDIN or $STDLIST have been redirected, because reading responses from a redirected $STDIN would be just another form of embedding passwords in command lines. Allowing passwords to be ommitted from job files will help reduce the risk of password exposure due to passwords embedded in job files.

ENHANCED PASSWORD PROMPTING EXAMPLES. The following logon password prompting examples assume the MANAGER.SYS user ID has both user and account passwords and that the SECURITY group has a group password. The following examples are assumed to have run under session number 72 logged on as the OPERATOR.SYS user ID using a terminal with the device name OPERTERM.

Job File :STREAM Example

```
:PRINT JOBFILE
!JOB JOBFILE,MANAGER.SYS,SECURITY
!SHOWME
!EOJ
:STREAM JOBFILE
ENTER ACCOUNT (SYS) PASSWORD:          (correct password supplied)
ENTER USER (MANAGER) PASSWORD:         (correct password supplied)
ENTER GROUP (SECURITY) PASSWORD:       (correct password supplied)
#J420
:
```

Interactive :STREAM Example

```
:STREAM
>!JOB INTERACT,MANAGER.SYS,SECURITY
ENTER ACCOUNT (SYS) PASSWORD:          (correct password supplied)
ENTER USER (MANAGER) PASSWORD:         (correct password supplied)
ENTER GROUP (SECURITY) PASSWORD:       (correct password supplied)
>!SHOWME
>!EOJ
#J421
>:
:
```

Job Authentication Failure Example

```
:STREAM JOBFILE
ENTER ACCOUNT (SYS) PASSWORD:          (incorrect password supplied)
ENTER ACCOUNT (SYS) PASSWORD:          (incorrect password supplied)
ENTER ACCOUNT (SYS) PASSWORD:          (incorrect password supplied)
INCORRECT PASSWORD.  (CIERR 1441)
:
```

On the system console the message **INVALID PASSWORD FOR "!" DURING LOGON ON LDEV #\. (js 65)** is displayed three times — once for each incorrect password. The user's logon information [jsname],user.account,group and the ldev of the user's logon device are substituted into the console message.

Interactions with Other Features

Existing password prompting will use the new password prompts which include the user's user, account, or group name. For example for the command :HELLO MANAGER.SYS the new prompt will be ENTER USER (MANAGER) PASSWORD rather than the old ENTER USER PASSWORD prompt.

The password prompting for the :CHGROUP command will be made consistent with other password prompting. An INCORRECT PASSWORD (CIERR 1441) error message will no longer be displayed on the user's terminal for the first two incorrect attempts for each password. This error message was not displayed during other password prompting dialogues. The present inconsistent behavior was the result of a programming defect in the STARTLOGON module.

If the Stream Initiation log type is enabled, a log record will be logged for all successful batch submissions.

Job Password Prompting Limitations

Although job password prompting can reduce the need to embed passwords in job files it does not eliminate this need. Jobs which stream other jobs must still contain embedded passwords. Job password prompting also does not address the desire to limit knowledge of authentication passwords to the user responsible for a user ID. Users must reveal their authentication passwords to other users if they wish to allow other users to submit jobs on their behalf. Future MPE XL products may provide a more secure alternate batch policy which addresses these issues.

STREAMJOB

SYNTAX

```
procedure STREAMJOB(
            var streamparms : p_in_buf_type; {input}
            var cierror     : integer;       {output}
            var jobnumber   : integer        {output}
            );
```

DESCRIPTION

Streamjob provides a supported interface for privilege-mode programs executing in either native mode or compatibility mode to stream jobs or data without being subject to job ID authentication or group access checks. All :JOB and :DATA command parameters passed to **streamjob** except passwords will be checked for validity. Passwords should not be embedded in these :JOB and :DATA commands, but will be ignored if present.

PARAMETER DEFINITIONS

streamparms — A string containing :STREAM command parameters terminated by a carriage return. The syntax of this string is the same as the :STREAM command's syntax:

```
[jobfile],[char] [ [;AT=hh:mm] [ [;DATE=mm/dd/yy]  ]   ]
                                [;DAY={dow|dom|dem}
                       [;IN=[days [, [hours] [, minutes]]]]
```

cierror — The CI error/warning number is returned in this parameter if an error occurs, otherwise zero is returned. **Streamjob** will return a subset of the errors returned by the :STREAM command. Errors relating to invalid passwords or lack of authorization should never be returned by **streamjob**.

jobnumber — The job number is returned in this parameter if the job is successfully :STREAMed, otherwise zero is returned.

ADDITIONAL DISCUSSION

Calling this interface a privilege mode program effectively satifies job ID authentication and logon group access checks without obtaining the authentication information required by the FOS batch submission policy. Since privilege mode programs have the capability to look up this information, this routine does not grant any additional capabilities to privilege mode programs. Privileged applications calling this

routine are responsible for enforcing their own batch submission policy in a manner which ensures this authority is not abused by unauthorized users.

AUTOINST will probably be the first application to use this interface. By using this interface AUTOINST will be able to successfully execute even when installation user IDs have been assigned passwords.

WARNING

This interface should NOT be documented in user manuals and should only be used by internal HP software engineers when other batch submission authorization methods are incapable of meeting the needs of an HP product. This interface should not be used outside of HP since it is not an architected interface. Use of this interface should be communicated to the MPE XL lab (CSY/MXO/CBL) to determine applicability and to allow dependency tracking.

INTERACTIONS WITH OTHER FEATURES

As previously mentioned, the **streamjob** interface bypasses all batch submission policy requirements except for validity checks on the user identity and syntactic checks on the content of the :STREAM command.

If Stream Initiation logging is enabled, a log record will be logged for all successful invocations of the **streamjob** routine.

CAPABILITIES REQUIRED

Privileged Mode (PM) is required.

Job Security Master Table

A new Job Security Table (JSEC) is created to keep job/session security information. JSEC is a system table similar to JMAT (as opposed to a job table like JDT or JIT). There is one JSEC per system; the DST for JSEC is DST #61.

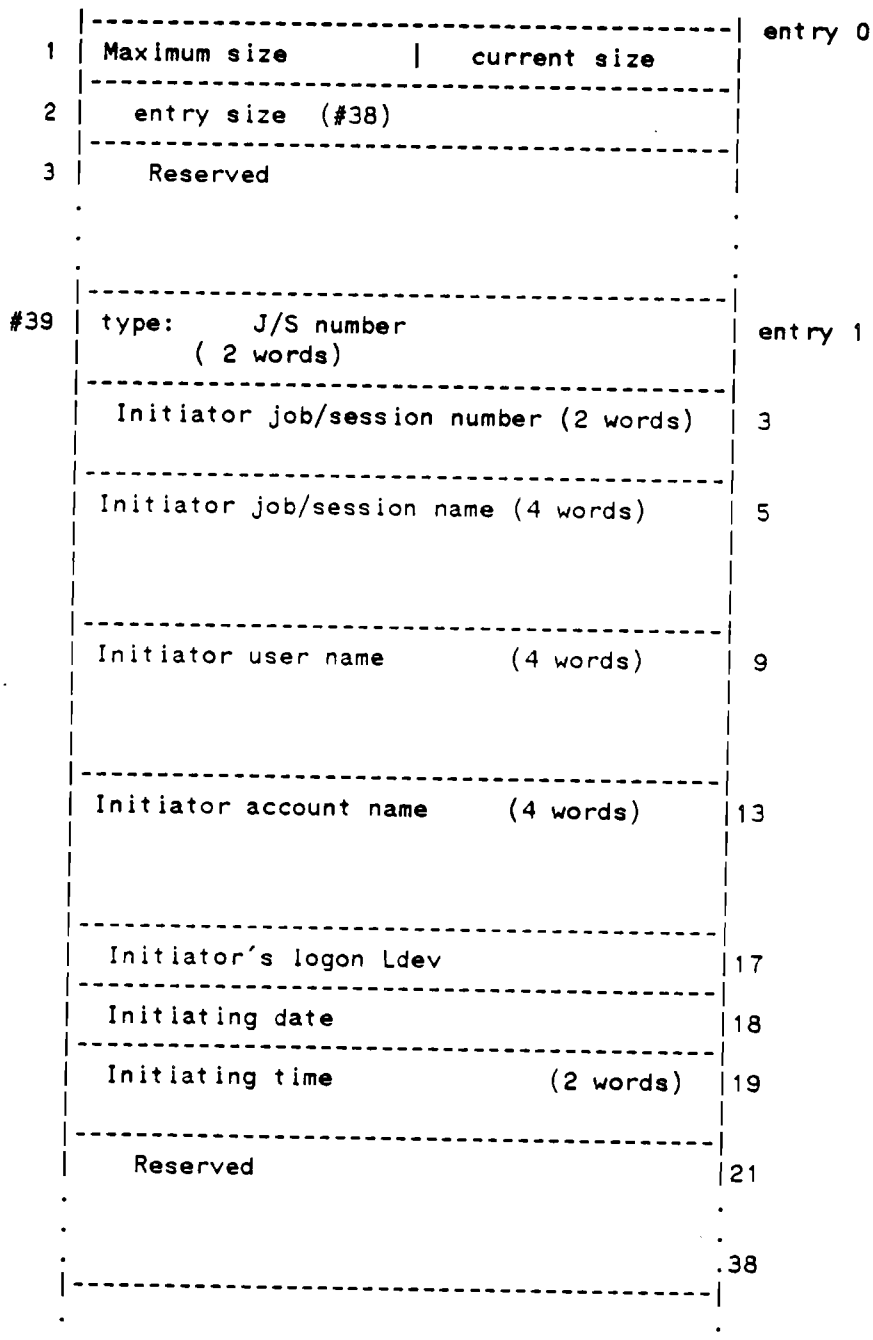
JSEC is the same entry size and total size as JMAT. The JMAT index is used to allocate, access and deallocate JSEC entries to individual jobs/sessions. CM'INITIAL in MPE XL creates or recovers JSEC the same way it does JMAT.

Note: In MPE XL, JMAT is currently a compatibility mode DST and has the same size/format as MPE V/E JMAT. JSEC, which can be viewed as a logical extension of the JMAT to hold security information, also has the same format in both operating systems.

JSEC Table Format

Like the JMAT, JSEC has a 38-word header and 38-word entries. The first two CM words of JSEC header is the same as JMAT. The rest of the header is reserved for the future use. For now, the only information contained in each entry is the initiator information. This information, together with the job number, takes 20 words. The rest is reserved. The format of JSEC table in pictorial form is on next page.

JSECDST Table (DST #61)



Creation of JSEC DST

JSEC (DST# 61) is created by CM'INITIAL at startup time. In START RECOVERY, the old copy from previous execution cycle (before SHUTDOWN or a system abort) is re-used so that information can be recovered. A fresh copy is created on any other type of start-up.

To be consistent with the way JMAT is created in MPE XL, the JSEC DST is preserved on disk in the form of a file: DSTJSEC.PUB.SYS (JMAT is contained in the file DSTJMAT.PUB.SYS).

When the DST is created, CM'INITIAL will simply "wrap" the file around the DST disk space, making the file data itself the disk copy of the DST. This is a technique already employed in MPE XL for preserving the permanent DSTs such as JMAT, WELCOME1, WELCOME2, RIN, ...

Below is the algorithm for JSEC creation code in CM'INITIAL.

```
IF not start_recovery THEN
    purge DSTJSEC.PUB.SYS;
  IF DSTJSEC not there THEN
    build DSTJSEC.PUB.SYS (size = JMAT max size);
  Create DST# 61 with size = JMAT current size
                 and disk addr = DSTJSEC file data;
  Copy first 2 CM words of JMAT to JSEC;
  Protect DSTJSEC.PUB.SYS to prevent purging;
```

Accessing JSEC entries

The JMAT SIR is used to lock the JSEC. This should not present any problem because the JSEC is of the same nature as the JMAT. In addition, it would not significantly increase the duration the JMAT SIR is held, because (1) accessing to the JSEC does not happen very frequently, and (2) the duration should be quite short, only during a data segment move.

Only job/session related security code will access JSEC entries. At this point, since the entry contains only job initiator information, two security procedures are provided to write and read this information: PUT_JINIT_INFO and PRINT_JINIT_INFO.

SECURITY AUDITING

The following security auditing enhancements will be implemented on the system logging facility:

- logging of password changes
- logging of system logging configuration
- logging of restore
- logging of printer access failure
- logging of ACD changes
- logging of stream initiation
- logging of user logging
- logging of process initiation
- auditability by named user

With these enhancements to auditing, system log files will be filling up faster. This will require more attention to archive old log files.

SYSGEN will show system logging configuration as:

configurable item	max	min	current
# of user logging processes	64	2	64
# users per logging process	256	1	128

system log events	event #	status
System logging enabled	100	OFF
System up record	101	ON
Job initiation record	102	OFF
Job termination record	103	OFF
Process termination record	104	OFF
File close record	105	OFF
System shutdown record	106	ON
Power failure record	107	ON
Spooling log record	108	OFF
I/O error record	111	ON
Physical mount/dismount	112	OFF
Logical/mount/dismount	113	OFF
Tape labels record	114	OFF
Console log record	115	ON
program file event	116	ON
New commercial spooling	120	ON
Architected interface	130	ON
Password changes	134	ON
System logging configuration	135	ON
Restore logging	136	ON
Printer access failure	137	ON
ACD changes	138	ON
Stream initiation	139	ON
User logging	140	ON
Process creation	141	ON
Chgroup record	143	ON
File open record	144	ON
Maintenance request log	146	OFF
diagnostic information record	150	ON
high priority machine check	151	OFF
low priority machine check	152	OFF
CM file close record	160	OFF

Logging of Password changes

Passwords can be changed by MPE commands, the security configuration utility, and other HP utility programs. This feature allows the system manager to audit the changing of passwords via the MPE XL commands (:ALTACCT, :ALTGROUP, :ALTUSER, and :PASSWORD) and the restore utility.

A new log type 134 will be added to the system logging facility. The MPE operating system and LOGTOOL will recognize this new log event. It can be enabled by SYSGEN followed by a START NORECOVERY. This event is initially disabled.

System logging will record when a user, group or account password is changed via MPE commands or other HP utility programs. The log record will contain the following information:

Header:

- record type
- record length
- timestamp
- job/session number
- PIN

Log information:

- identification of user who changed password (includes job/session name, user name, group name, account name)
- identification of user whose password was changed. (includes account name if account password changed; group name if group password changed; user and account names if user password changed)
- input logical device number from which the password was changed
- program file name from which password change was executed
- type changed (1 User; 2 Group; 4 Account)

For example, JOHN.PAYROLL,DOE with job/session name JREPORT, successfully changed account password for PAYROLL account via the command executor. The change was made from ldev 21.

LOGTOOL will format the following layout after the standard header:

```
TARGET USER:                        TARGET GROUP:
TARGET ACCOUNT:     PAYROLL         TYPE CHANGED:     ACCOUNT
LDEV:               21
EXECUTED FROM:      CI.PUB.SYS
USER:               JOHN            GROUP:            DOE
ACCOUNT:            PAYROLL         JSNAME:           JREPORT
```

The following is the password changes logging record format:

length in word	bit 0 15
(1)	Record type = 134
(1)	Record length
(1)	PIN
(3)	Time stamp
(2)	Job type / job number
(8)	TARGET USER NAME
(8)	TARGET GROUP NAME
(8)	TARGET ACCOUNT NAME
(1)	TYPE CHANGED
(1)	INPUT LDEV NUMBER
(25)	EXECUTED FROM
(3)	Reserved
(8)	USER NAME
(8)	GROUP NAME
(8)	ACCOUNT NAME
(8)	JOB/SESSION NAME

Logging of System Logging Configuration

The run time system logging configuration can be altered at boot time using a configuration file created by SYSGEN. Additionally, the security configuration utility will be able to alter the. This feature will provide the audit trial of the logging configuration changes.

A new log type 135 will be added to the system logging facility. It can be enabled by SYSGEN followed by a START NORECOVERY or by the security configuration utility without a subsequent START. This event is initially enabled.

The MPE operating system and LOGTOOL will recognize this new log event. When this feature is enabled, system logging configuration will be recorded when the system is started up or the configuration is altered via security configuration utility.

The log record will contain the following information:

Header:

- **record type**
- **record length**
- **timestamp**
- job/session number
- PIN

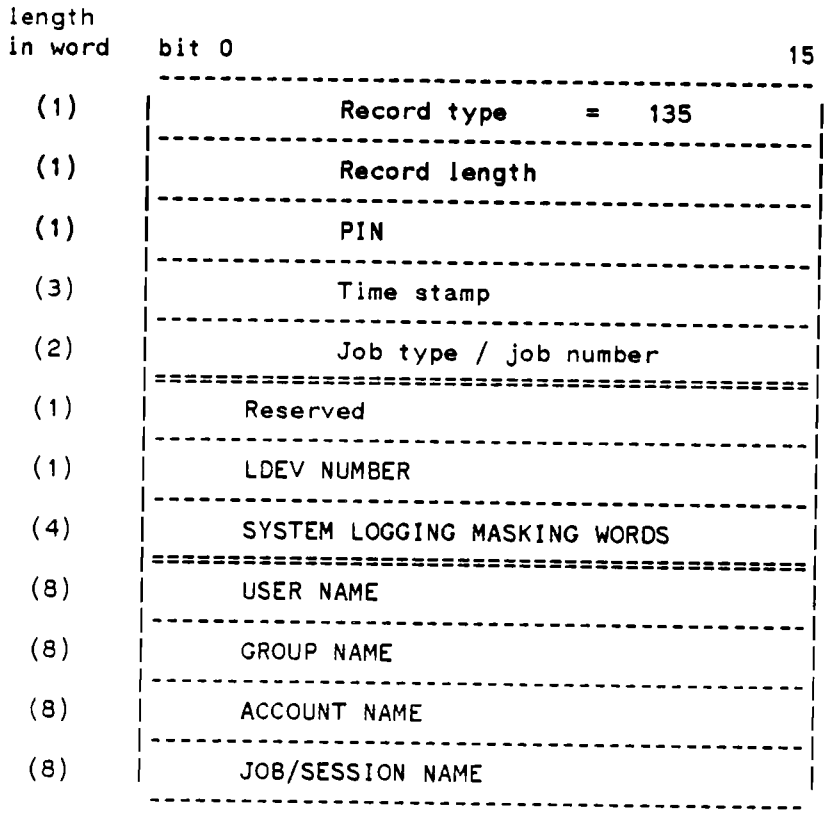
Log information:

- identification of the user who make changes to the configuration (includes job/session name, user name, group name, account name)
- logical device number
- new system logging configuration

For example, user CONFIG,MANAGER.SYS,PUB changed the system logging configuration via security configuration utility from ldev 20. The system log types 100, 101, 102, 111, 151, and 152 are enabled. LOGTOOL will format it as:-

```
LOGICAL DEVICE:          20
SYSTEM LOG CONFIG:
LOG FAILURE:             ON          SYSTEM UP:              ON
JOB INITIATION:          ON          JOB TERMINATION:        OFF
PROCESS TERMINATION:     OFF         NM FILE CLOSE:          OFF
SHUTDOWN:                OFF         POWER FAILURE:          OFF
SPOOLING:                OFF         I/O ERROR:              ON
PHY. MOUNT/DISMOUNT:     OFF         LOG. MOUNT/DISMOUNT:    OFF
TAPE LABELS:             OFF         CONSOLE LOG:            OFF
PROGRAM FILE EVENT:      OFF         NCS LOGGING:            OFF
AIF LOGGING:             OFF         PASSWORD CHANGE:        OFF
SYS LOG CONFIG:          OFF         RESTORE:                OFF
PRINTER ACCESS:          OFF         ACD CHANGE:             OFF
STREAM INITIATION:       OFF         USER LOGGING:           OFF
PROCESS CREATION:        OFF         SECURITY CONFIG:        OFF
CHGROUP:                 OFF         FILE OPEN:              OFF
COMMAND LOGGING:         OFF         AUTO-DIAG/SUM:          OFF
HPMC:                    ON          LPMC:                   ON
CM FILE CLOSE:           OFF
USER:                    MANAGER     GROUP:                  PUB
ACCOUNT:                 SYS         JSNAME:                 CONFIG
```

The following is the system up log record format:



Logging of Restore

Files can be restored from tape or serial disc to the system. The creation of such files via restore needs to be auditable.

A new log type 136 will be added to the system logging facility. It can be enabled by SYSGEN followed by a START NORECOVERY or by the security configuration utility without a subsequent START. This logging type is initially disabled.

The MPE operating system and LOGTOOL will recognize this new log event. If this feature is enabled, the system logging facility will log restore of files.

The log record will contain the following information:

Header:

- record type
- record length
- timestamp
- job/session number
- PIN

Log information:

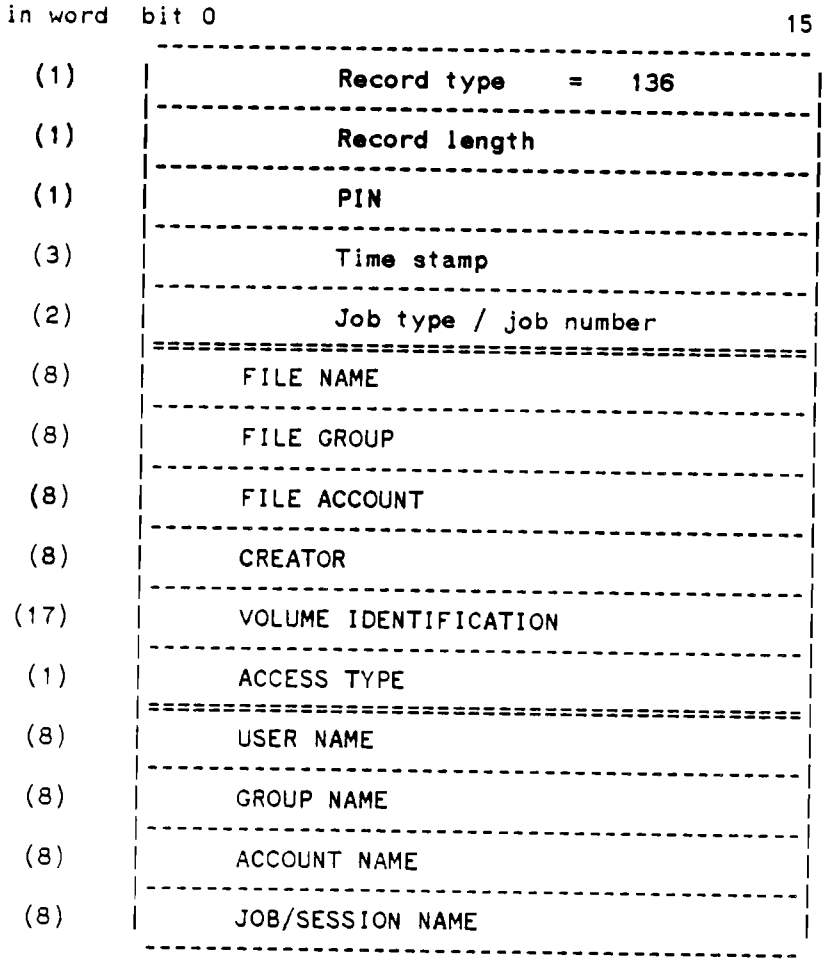
- identification of the user who restored a file (includes job/session name, user name, group name, account name)
- name of the file which was restored (includes file name, group name, account name)
- name of the creator of the file
- disc volume identification where the file will reside. It consists of volume type and volume type name. (volume type: 0 = volume name, 1 = volume class name, 2 = volume set name)
- access type: 1 = Restores a new file 2 = Replaces an existed disc file

For example, a user, JOHN.PAYROLL,DOE with job/session name as JREPORT, restored a new file FTEST.TESTGP.PAYROLL on a volume set called MY_TEST_VOL_SET. The file creator was DOLE.

LOGTOOL will format the following layout after the standard header:

```
FILE NAME:          FTEST          FILE GROUP:          TESTGP
FILE ACCOUNT:       PAYROLL        CREATOR:             DOLE
ACCESS TYPE:        1
VOLUME ID:          MY_TEST_VOL_SET
USER NAME:          JOHN           USER GROUP:          DOE
USER ACCOUNT:       PAYROLL        JOB/SESSION NAME:    JREPORT
```

The following is the Restore log record format:



Logging of Printer Access Failure

This feature allows the system manager to audit failures in attaching spoolfiles to printers. This does not include creating new spoolfiles which are logged by FOPEN.

A new log type 137 will be added to the system logging facility. It can be enabled by SYSGEN followed by a START NORECOVERY or by the security configuration utility without a subsequent START. This event is initially disabled.

The MPE operating system and LOGTOOL will recognize this new log event. When this feature is enabled, failures to attach spoolfiles to printers will be logged.

The log record will contain the following information:

Header:

- record type
- record length
- timestamp
- job/session number
- PIN

Log information:

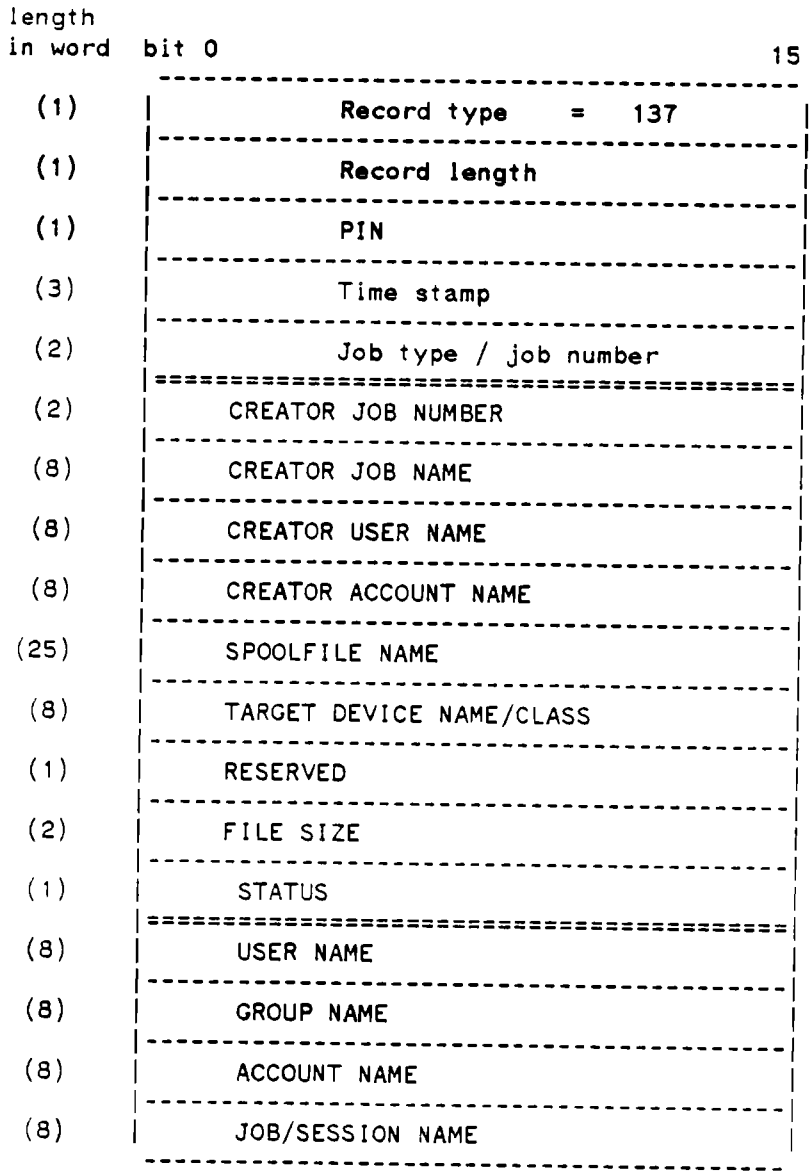
- identification of the user who owns the output spoolfile (includes job number, job name, user name, account name)
- spoolfile name
- target device name/class
- number of records in the output spoolfile
- failure flag: 1 = internal failure of access check,
  2 = security violation

For example, a user, JOHN.PAYROLL,DOE with job number #J12 and job name as JREPORT, spooled an output file FNAME.TEST.PAYROLL with file size of 200 records to a printer whose target device class name was LP. It failed in device ACD security check.

LOGTOOL will format the following layout after the standard header:

```
USER:              JOHN              ACCOUNT:            PAYROLL
J/S TYPE  :        JOB               JOB NUMBER:         12
JOB NAME:          JREPORT
SPOOLFILE NAME:    FNAME.TEST.PAYROLL
FILE SIZE:         200               TARGET DEVICE:      LP
FAILURE FLAG:      Security violation
USER NAME:         JOHN          USER GROUP:         DOE
USER ACCOUNT:      PAYROLL       JOB/SESSION NAME:   JREPORT
```

The following is the printer access failure logging record format:



Logging of ACD changes

ACDs can be changed by MPE commands and intrinsic calls. This feature allows the system manager to audit both types of these events.

A new log type 138 will be added to the system logging facility. The MPE operating system and LOGTOOL will recognize this new log event. It can be enabled by SYSGEN followed by a START

NORECOVERY or by the security configuration utility without a subsequent START. This event is initially disabled.

System logging will record when ACDs are changed (created, deleted, copied, or modified) via MPE commands and intrinsic calls. The log record will contain the following information:

Header:

- **record type**
- **record length**
- timestamp
- job/session number
- PIN

Log information:

- identification of the user who changed ACD (includes job/session name, user name, group name, account name)
- object type and object name whose ACD was changed
- object type and object name from where ACD was copied
- type of change to the ACD: create, add pair, replace pair, copy, delete pair, or delete
- program file name from which ACD change was executed
- status returned (HPE status)

For example, a user JOHN.PAYROLL,DOE with job/session name as JREPORT, successfully created an ACD for a file FTEST.TESTGP.PAYROLL from the command executor.

LOGTOOL will format the following layout after the standard header:

```
TARGET OBJECT:      FTEST.TESTGP.PAYROLL
SOURCE OBJECT:
FUNCTION:           CREATE
EXECUTED FROM:      CI.PUB.SYS
STATUS:             Successful
USER:               JOHN            GROUP:      DOE
ACCOUNT:            PAYROLL         JSNAME:     JREPORT
```

The following is the ACD changes logging record format:

length in word	bit 0 15
(1)	Record type = 138
(1)	Record length
(1)	PIN
(3)	Time stamp
(2)	Job type / job number
(25)	TARGET OBJECT NAME
(25)	SOURCE OBJECT NAME
(4)	FUNCTION
(25)	EXECUTED FROM
(2)	STATUS
(8)	USER NAME
(8)	GROUP NAME
(8)	ACCOUNT NAME
(8)	JOB/SESSION NAME

Logging of Stream Initiation

This feature enables the system manager to know who streams or tries to stream a job, and when and where it occurs.

A new log type 139 will be added to the system logging facility. The MPE operating system and log file report generator, LOGTOOL will recognize this new log event. It can be enabled by SYSGEN followed by a START NORECOVERY or by the security configuration utility without a subsequent START. This event is initially disabled.

When this feature is enabled, any user streaming a job will have the event logged. The new log record will contain the following information:

Header:

- record type
- record length
- timestamp
- job/session number
- PIN

Log information:

- identification of the user who streamed the job (includes job/session name, user name, group name, account name)
- name of the job file that was streamed (includes file name, group name, account name)
- logical device number where the job was streamed
- identification assumed by the job (includes j/s number, user name, group name, account name)
- time that the job was scheduled to be launched
- name of the job streamed

For example, a user, with user name JOHN, account name PAYROLL, group name DOE, and job/session name JREPORT, streamed a job file JTEST.TESTGP.TESTACCT which logged on as #J12 QAMGR.QAACCT,QAGP with a job name as JOBTEST from ldev 21 and an input spool id #15. The job is scheduled to be launched at 10:00 am, Wednesday, Sept. 30, 1989.

LOGTOOL will format the following layout after the standard header:

```
INPUT LDEV:          21
JOB FILE NAME:       JTEST.TESTGP.TESTACCT
LOGON J/S TYPE:      JOB                  LOGON J/S NUMBER:   12
LOGON USER:          QAMGR                LOGON GROUP:        QAGP
LOGON ACCOUNT:       QAACCT               LOGON JOB NAME:     JOBTEST
INPUT SPOOLFILE ID:  15
SCHEDULE DATE:       WED, SEP 30, 1989    SCHEDULE TIME:      10:00 AM
USER:                JOHN                 GROUP:              DOE
ACCOUNT:             PAYROLL              JSNAME:             JREPORT
```

The following is the stream initiation logging record format:

length in word	bit 0 15
(1)	Record type = 139
(1)	Record length
(1)	PIN
(3)	Time stamp
(2)	Job type / job number
(1)	INPUT LDEV
(25)	JOB FILE NAME
(2)	JOB LOGON J/S NUMBER
(8)	JOB LOGON USER
(8)	JOB LOGON GROUP
(8)	JOB LOGON ACCOUNT
(8)	JOB NAME
(2)	INPUT SPOOLFILE ID
(1)	SCHEDULED DATE
(2)	SCHEDULED TIME
(8)	USER NAME
(8)	GROUP NAME
(8)	ACCOUNT NAME
(8)	JOB/SESSION NAME

Logging of User Logging

This feature enables the system manager to determine who accesses or tries to access the user logging facility by logging all OPENLOG and CLOSELOG intrinsic calls.

A new log type 140 will be added to the system logging facility. This new log record will be recognized by the MPE operating system and log file report generator LOGTOOL. It can be enabled by SYSGEN

followed by a START NORECOVERY or by the security configuration utility without a subsequent START. This event is initially disabled.

When it is enabled, all OPENLOG and CLOSELOG intrinsic calls will be logged. The new log record will have the following information:

Header:

- record type
- record length
- timestamp
- job/session number
- PIN

Log information:

- identification of the user who called OPENLOG or CLOSELOG (includes job/session name, user name, group name, account name)
- name of the program file from where OPENLOG or CLOSELOG was called
- intrinsic that the user called, either OPENLOG or CLOSELOG
- logging index (its value is returned from OPENLOG. It identifies the access to the user logging facility)
- logging identification
- mode of the intrinsic call, either 0 for WAIT or 1 for NOWAIT
- status of the intrinsic call

For example, a user JOHN.PAYROLL,DOE with job/session name as JREPORT, ran a program FTEST.TESTGP.TESTACCT which called the OPENLOG intrinsic. The intrinsic call had index 1000, logid JOHNID, mode 0 and there was no error returned from the call.

LOGTOOL will format the following layout after the standard header:

```
PROGRAM FILE NAME:  FTEST.TESTGP.TESTACCT
INTRINSIC:          OPENLOG          INDEX:      1000
LOG ID:             JOHNID           MODE:       WAIT
STATUS:             0
USER:               JOHN             GROUP:      DOE
ACCOUNT:            PAYROLL          JSNAME:     JREPORT
```

The following is the user logging log record format:

length in word	bit 0 — 15
(1)	Record type = 140
(1)	Record length
(1)	PIN
(3)	Time stamp
(2)	Job type / job number
(25)	PROGRAM FILE NAME
(4)	INTRINSIC
(2)	INDEX
(4)	LOGID
(1)	MODE
(1)	STATUS
(8)	USER NAME
(8)	GROUP NAME
(8)	ACCOUNT NAME
(8)	JOB/SESSION NAME

NOTE

The LOG ID field in the log record will be "XXXXXX" for CLOSELOG intrinsic when the index is bad.

Logging of Process Creation

This feature provides information for tracing the creation of a process on the system.

A new log type 141 will be added to the system logging facility. It can be enabled by SYSGEN followed by a START NORECOVERY or by the security configuration utility without a subsequent START. This event is initially disabled.

When it is enabled, the system logging facility will log all process creations. The new log record will have the following information:

Header:

- record type
- record length
- timestamp
- job/session number
- PIN

Log information:

- identification of the user who initiated the process (includes job/session name, user name, group name, account name)
- process identification (PID) initiated
- parent PID
- priority of the process
- process space ID
- program name that the process was initiated from
- capabilities of the created process
- native mode heap size in bytes

For example, a user JOHN.PAYROLL,DOE with job/session name as JREPORT, ran a program FTEST.TESTGP.TESTACCT with PID 2300000002. The program initiated a process with PID 3300000001, priority 130 and space ID 556. The process created had capabilities of IA,BA,PH,PM and had NM heap size 200000 bytes.

LOGTOOL will format the following layout after the standard header:

```
PROGRAM FILE NAME:  FTEST.TESTGP.TESTACCT
PID INITIATED:      3300000001      PRIORITY:           130
SPACE ID:           556             PARENT PID:         2300000002
NM_HEAP_SIZE:       200000
PROCESS CAP BA:        YES          PROCESS CAP IA:     YES
PROCESS CAP PM:        YES          PROCESS CAP MR:     NO
PROCESS CAP DS:        NO           PROCESS CAP PH:     YES
USER:               JOHN            GROUP:              DOE
ACCOUNT:            PAYROLL         JSNAME:             JREPORT
```

The following is the process initiation log record format:

length in word	bit 0 — 15
(1)	Record type = 141
(1)	Record length
(1)	PIN
(3)	Time stamp
(2)	Job type / job number
(25)	FILE NAME
(1)	RESERVED
(2)	PRIORITY
(2)	PROCESS SPACE ID
(4)	PARENT PID
(2)	NM_HEAP_SIZE
(2)	PROCESS CAPABILITIES MASK
(8)	RESERVED
(8)	USER NAME
(8)	GROUP NAME
(8)	ACCOUNT NAME
(8)	JOB/SESSION NAME

System logging mask:

User Attributes
bit capability
0 - SM
1 - AM
2 - AL
3 - GL
4 - DI
5 - OP

Program/Group Attributes
bit capability
23 - BA
24 - IA
25 - PM
28 - MR
30 - DS
31 - PH

File access attributes
6 - CV
7 - UV
8 - LG
9 - SP
10 - PS
11 - NA
12 - NM
13 - CS
14 - ND
15 - SF

Auditability by Named User

This feature provides the ability to selectively audit the actions of one or more users based on individual identity. LOGTOOL will allow users of the utility to format log records selected by user identifications.

Three optional parameters: JSNAME, USER and ACCOUNT will be added to the LIST command in LOGTOOL. The syntax will be:

```
LIST LOG=<log list>
     [;JSNAME=<job/session name>]
     [;USER=<user name>]
     [;ACCOUNT=<account name>]
```

The input should not be longer than 80 characters. The default value of these new parameters are @.

For example, to LIST log records from log file number 1 to 5, with log type 142 and user identification JTEST,JOHN.PAYROLL, the LIST command will be:

>LIST LOG=1/5;TYPE=142;JSNAME=JTEST;USER=JOHN;ACCOUNT=PAYROLL

To select log records from log file number 1 to 5, with log type 142 and user identification @,@.PAYROLL, the LIST command will be:

>LIST LOG=1/5;TYPE=142;ACCOUNT=PAYROLL

To allow LOGTOOL selecting security relevant log records based on the user identification, all security relevant log records will contain user identification entries (JSNAME, USER, and ACCOUNT). Those security relevant log events are:

- job initiation (type 102)
- job termination (type 103)
- process termination (type 104)
- file close (type 105, 160)
- spooling log (type 108)
- physical mount/dismount (type 112)
- logical mount/dismount (type 113)
- tape labels (type 114)
- console log (type 115)
- program file event (type 116)
- New Commercial Spooling (type 120)
- Architecture Interface (type 130)
- password change (type 134)
- system logging configuration (type 135)
- restore (type 136)
- printer access failure (type 137)
- ACD change (type 138)
- stream initiation (type 139)
- user logging access (type 140)
- process initiation (type 141)
- security monitor configuration change (type 142)
- change group (type 143)
- file open (type 144)
- command logging (type 145)

The following existing log records will be appended with user identification entries:

- job termination (type 103)
- process termination (type 104)
- physical mount/dismount (type 112)
- tape labels (type 114)
- console log (type 115)
- program file event (type 116)
- New Commercial Spooling (type 120)
- change group (type 143)

MULTIPROCESSOR SYSTEM

SECTION 4

INTRODUCTION

This article focuses on providing a brief overview of functional areas and data structures that have changed for multiprocessor support on MPE XL. (See additional training section.)

A multiprocessor system contains two or more processors. The multiprocessor implementation takes advantage of Precision Architecture-RISC (PA-RISC) to provide incremental performance upgrades. The multiprocessor system can be defined by the following hardware and software characteristics:

- **Symmetric.** Each of the processors has equal capability. Tasks can be assigned to both processors without restrictions.
- **Tightly Coupled.** All processors have access to a common shared memory.
- **Shared I/O.** Access to any I/O device is allowed from any processor.
- **Single Integrated Operating System.** There is a single operating system in control of all hardware and software.

Our multiprocessor implementation has two minor exceptions to the symmetric definition. One, during I/O configuration, channels are assigned to each processor, resulting in directed I/O completion interrupts. A directed interrupt is defined as a channel which interrupts only its assigned processor when an I/O operation completes. Two, timer interrupts are directed to the monarch processor and portions of timer services relating to timer notification are handled only by the monarch. See monarch definition below.

HARDWARE OVERVIEW

The multiprocessor hardware implementation provides a tightly-coupled hardware organization with two symmetric processors. The processors reside on the System Main Bus (SMB). Each processor has its own cache and Translation Look-Aside Buffer (TLB). PA-RISC specifies that the hardware will ensure data consistency across the multiple caches and TLBs. Thus, the software views the hardware as if there were a single cache and TLB. Maintaining cache coherency adds a level of hardware overhead that does not exist in a uniprocessor environment.

During the boot sequence all processors undergo a hardware self-test. In a multiprocessor environment, one of the processors is selected to become the "monarch"; this processor is responsible for the initial portion of the system boot sequence. The selection of the monarch is implementation dependent. This processor is always assigned logical processor ID of zero by MPE XL.

Once selected, the monarch processor is reponsible for the Initial System Load (ISL). ISL intiates the MPE XL system launch code that begins the operating system initialization. Later in the system startup sequence, the second processor is automatically initialized by MPE XL. Messages are sent to the console for successful processor invokation. If a processor fails, messages are sent to the console and to the system log file. The system log record identifies the logical processor id, the hard physical address, and an error status.

If a processor fails because of technical board lockout implemented by SW_ID, the system halts with the same display as a uniprocessor system. If a processor fails for a reason other than technical board lockout, the system will boot on the remaining good processor. This is known as booting in "degraded mode"; booting in degraded mode has no impact on the system integrity, although performance will be degraded. The system will operate at the performance level of a uniprocessor system.

It may be possible to disable a processor during bootup if the processor is suspected of having a hardware problem. If implemented, this would occur as an option on the "Enter boot path command". The DISABLE command followed by the physical slot position of the processor card will disable the processor. To obtain a display of command options enter a question mark at the "Enter boot path" prompt.

SOFTWARE OVERVIEW

The effects of a multiprocessor environment on the user are mostly transparent. However, changes to low level internal services and data structures are required to support a multiprocessor implementation. The rest of this article provides an overview of these changes.

Simultaneous Interrupt Processing

On MP, each processor is capable of servicing interrupts independently. There are separate interrupt vector address tables (IVA) and interrupt control stacks (ICS) for each processor. Also, a new fast interrupt mechanism supported by the System Monitor provides inter-processor communication for specific events such as clock synchronization and shutdown notification.

Timer Services

At system startup time, the monarch is assigned to be the Timer Processor. It generates "heart beats" for the entire system. A heart beat is a timer interrupt that occurs every 8 milliseconds. The System Monitor manages the heart beat timer. This timer is used to support the implementation of several system functions:

- maintain timer counters to manage the timer request list
- maintain dispatcher counters to support timeslice events
- maintain the activity counters for updating the activity display and performing clock synchronization

Dispatcher

In a multiprocessor environment it is possible to have more than one currently active process. Processes can be launched on any processor, and throughout their lives, may migrate between processors. For the most part, process scheduling rules remain the same. The dispatcher ensures the two highest priority ready processes will be selected to run.

The dispatcher can run concurrently on both processors since each processor has its own Interrupt Control Stack (ICS). However, there is a single shared ready queue containing a list of ready processes linked in priority order. Synchronized access to this ready queue for concurrent executions of the dispatcher is maintained.

A new tune feature is available for timeslice management of processes running in the CS, DS, and ES scheduling queues. See the Communicator article on the :TUNE command for more information.

Port ICS Priority Management Changes

ICS priority management of procedure servers has changed. A new mechanism for scheduling deferred ports has been implemented. ICS procedure servers are deferred if the server is already active, the subqueue is disabled, or an explicit request is given. These deferred requests will be linked into a queue for each processor and serviced during IEXIT processing.

Semaphores

The CB_Lock routines have been changed to support new semaphore record structures and types. These routines help support additional synchronization requirements for low level services.

Synchronization Issues

Alternate methods of synchronization are required by multiprocessor software. Prior to MP, disabling interrupts of disabling the dispatcher guaranteed that a process or service was the only one touching an area of internal data. On MP, disabling interrupts or the dispatcher will only guarantee that a process will not get preempted or timesliced. There is no control over the other processor or the internal data areas it can touch. Therefore, alternate sychronization methods were added throughout low level services. Examples of previously non-concurrent low level services are the Dispatcher, Memory Manager, System Monitor, and low level trap support.

For example, if the dispatcher is running on each processor and both processors need access to the ready queue; one processor is now required to spin (loop) while the other processor is accessing the ready queue. The CB_LOCK routines utilize a 'spin word' to accomplish this. A spin word is a 16-byte aligned word of memory that can be interrogated by the Load and Clear Word (LDCW) semaphore instruction. The LDCW instruction performs an atomic load and clear word operation. (See the Precision Architecture and Instruction Set Reference Manual for details). Mutually exclusive access to shared internal data areas has been implemented throughout the low level services of the operating system.

This type of synchronization does not come without a cost. System overhead has increased within the low level internal services of the system. This partially explains why there are not linear performance gains with an additional processor. Other areas of the operating system have been tuned to attempt to offset some of the multiprocessor overhead incurred by hardware and software.

Data Structure Changes

The multiprocessor implementation of MPE XL must support multiple simultaneous activities (i.e., multiple processes and/or interrupt processing). One obvious aspect of providing that level of activity is modification of the data structures that contain status and state information so that there is a clear division between data that is specific (private) to a processor and data that is global to the entire system.

The following is a list of some of the data structures which have changed:

- **System Globals.** Restructured into global and processor specific data (PSD) areas.

- **Real Globals.** Restructured into global and PSD areas.
- **Dispatcher Globals.** Restructured into global and PSD areas.
- **Interrupt Vector Address Table/Offsets.** One table per processor.
- **ICS.** Separate stack per processor.
- **MP Processor Array.** New structure contains one entry per processor.
- **Measurement Counter Table.** Separate counters per processor
- **Global Port Object.** Removal of ICS priority subqueues and cache.
- **Footprint Table.** The lower five bits of the timestamp contain the processor ID.
- **System Error Stack.** Addition of processor ID field.
- **Semaphore Record.** Significant changes to record structure and additional semaphore types.

New routines have been written to provide efficient access to PSD areas.

Dat/Debug

Debug remains a process level debugger. Therefore, similar debugging techniques apply in a multiprocessor environment as in a uniprocessor environment. DAT has been modified to support multiprocessor dump analysis. A new command, CPU, allows you to switch to another processor in order to examine a stack trace.

Impact on Privilege Mode Users

Outside of HP there are a number of privileged third party software packages that may reference the changed structures listed above. These products should look for dependencies on these structures. When possible, the privilege mode developers should be encouraged to use the Architected Interface Facility to access system tables. However, if additional internal information is required, requests should be directed to the CSY VAB Partnership Consulting Group previously known as TAC.

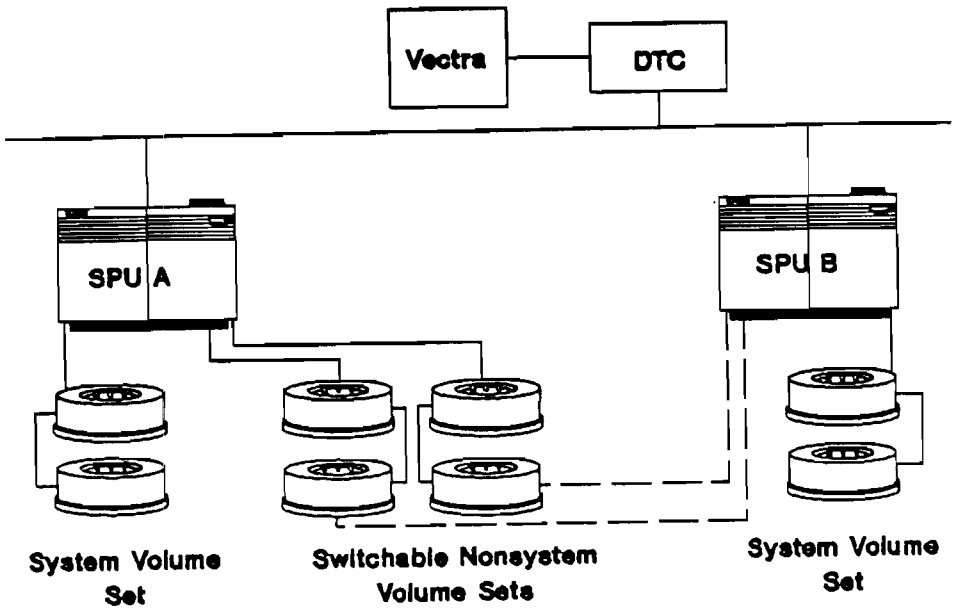
Additional Training

CSY Support is developing a three day video taped class to cover internal changes for multiprocessor (MP). The class materials can be ordered in December 1990. The ordering procedure for class materials will be announced in a CSY Newsletter.

SPU SWITCHOVER/XL

SECTION 5

SPU Switchover/XL is a new product designed to increase data availability by allowing the operator to switch applications and data from a failed SPU to a working SPU and continue processing. In order to utilize this software, extensive hardware is required. The figure below illustrates the necessary hardware configuration.



HARDWARE REQUIREMENTS

SPU Switchover/XL requires two HP 3000 Series 900 computers of comparable performance running MPE XL 3.0. These systems should have comparable performance so that either can undertake the processing of the other's applications.

As shown in the illustration above, there are one or more nonsystem volume sets (referred to as "switchable sets") that are connected to both SPUs. These switchable sets are made up of HP-FL disks and should hold all critical applications and data. Each switchable set has a "home" SPU where the set normally resides. When a switchable set mounts, its disks are locked so that only the SPU mounting the set can access the disks.

The remaining hardware requirement is a dedicated Vectra. This personal computer is needed to run the OpenView DTC Manager (HP D2355A). OpenView DTC Manager facilitates DTC switching.

SPU Switchover/XL and Mirrored Disks

Note that while mirrored disks are not a requirement for SPU Switchover/XL, customers may opt to use them with the product to provide another tier of defense for ensuring high data availability. Mirrored volume sets can be configured as switchable sets. The mirrored disks could provide the first line of defense for ensuring data availability; that is, if one drive failed, its partner would still be available. The

second line of defense would be handled by SPU Switchover/XL; if one SPU failed, its partner could take over.

INSTALLING AND CONFIGURING SPU SWITCHOVER/XL

The SPU Switchover/XL software must be installed on both SPUs in the SPU pair. When a customer purchases SPU Switchover/XL, the product is included on the SUBSYS tape and installed during the normal AUTOINST procedure of the update to release 3.0. The SPU Switchover/XL software includes a program file, SPUIDENT.MPEXL.SYS, which is invoked when any of the SPU Switchover/XL commands are issued.

An SPU switch over configuration file must be created before the product can be used. SYSGEN has been enhanced with a new first level command, SPU, that presents the user with a second level menu of SPU switch over configuration commands. The *SPU Switchover/XL User's Guide* (P/N3 6378-90001) contains a detailed description of how to set up an SPU switch over configuration.

Assume the two SPUs are called SPU A and SPU B as shown in the figure above. The user creates the SPU Switchover/XL configuration file on only one of these SPUs, lets say SPU A. The user must still invoke SYSGEN's SPU menu on SPU B in order to identify SPU B's partner as SPU A (refer to the manual for more information on the IDENTIFY command of SYSGEN's SPU menu). The SPU Switchover/XL configuration file will be transferred to SPU B when a SPUCONTROL SETUP is done on SPU A.

It is not necessary to make a System Load Tape (SLT) and do an update after creating the SPU Switchover/XL configuration file. It is not even necessary to reboot. The only time rebooting is necessary is when the SPU Switchover/XL configuration is not kept into the default SYSGEN configuration group; the SPUCONTROL SETUP command will look for the configuration file in this group.

Directory Preparation

The system volume set of each SPU must contain the accounting structure for all switchable sets, even those homed to the other SPU. The user must remember to set up the accounting structure on the partner SPU's system volume set. Otherwise when a switch over is needed, the accounting structure will not be in place and time delays will result.

Groups homed to a switchable set on one SPU should not be in use by a switchable set on the other SPU. For example, if a switchable set homed to SPU A has a DEPT47 group in its PAYROLL account, a switchable set homed to SPU B cannot have a DEPT47 group in its PAYROLL account. Otherwise, when SPU A is switched over to SPU B, the files in the DEPT47 group of the PAYROLL account on SPU A's switchable set would not be referenced since the system accounting structure on SPU B would think (and correctly so) that the DEPT47 group was homed to one of SPU B's switchable sets. It may be wise to avoid duplicating accounts on switchable sets homed to the different SPUs, in addition to avoiding duplicating groups.

DISK LOCKING

When an SPU is booted, it configures and locks each disk drive connected to it. It releases the locks for the drives not homed to it. Locking information is kept in the Mounted Volumes Table (MVT).

Disk locking is required to prevent one SPU from inadvertently corrupting data on a switchable volume set in use by the other SPU. It is accomplished by the Fiberlink Device Manager (FL DM) using the FLEX

command set. The FLEX command set allows an individual FL device to be locked by a host SPU (opcode #99 or $63). Once this is done, the SPU has exclusive access to the device until a device unlock is issued (opcode #107 or $6b). For more information on the FLEX command set, refer to the *FLEX Instruction Set Programming Manual* (P/NA-5959-3908-1).

SPU COMMUNICATION

The two partner SPUs communicate using the fiberlink path. This path is used to exchange information about locked disks, volume sets, configuration, and system state. Messages are passed through the fiber link connection from the A-Link card on one SPU to the A-Link card on its partner through the disk controller.

HANDLING FAILURES

When one system fails, the operator may either leave the system down or perform a switch over. Switch overs are not performed automatically. The operator must evaluate the situation and determine what action to take.

Switch Over

Following is an example of how an operator will perform a switch over from SPU A to SPU B.

1. SPU A fails to respond to a message sent from SPU B.
2. A message is displayed on the console of SPU B that SPU A may have suffered a hardware failure.
3. The operator acknowledges the message on SPU B and decides to begin a switch over.
4. The operator examines the load on SPU B to determine whether certain applications and/or sessions should be terminated in order to avoid performance problems when SPU A's applications are switched over.
5. The operator then issues the SWITCHOVER command on SPU B to perform the the switch over. SWITCHOVER will open SPU A's switchable sets on SPU B. The time needed for the switch over to complete depends on the number of volumes in SPU A's switchable sets. Transaction management (XM) recovery will be done on each switchable set; XM recovery takes an average of three minutes per volume.
6. The operator may invoke a command file on SPU B to perform recovery procedures and/or cleanup. Then the operator will restart SPU A's applications on SPU B.
7. The operator tells SPU A users to log on to SPU B and continue working.

A switch over may be desired even if neither SPU fails in order to perform system maintenance or to achieve a better load balance between the paired systems. In this case, steps one through three outline above would be replaced by the operator logging off applications and sessions on SPU A and VSCLOSing the switchable sets on that system.

Switch Back

Once SPU A has been repaired, the operator can decide to perform a switch back to move its applications back. There are five steps involved in a switch back:

1. The operator tells SPU A users logged onto SPU B to log off SPU B.
2. The operator VSCLOSEs SPU A's switchable sets which are currently open on SPU B.
3. The operator issues the SWITCHBACK command on SPU B to switch back SPU A's switchable sets.
4. When the SWITCHBACK is finished, the operator restarts applications on SPU A.
5. The operator tells SPU A users to log back on SPU A and continue working.

DATA STRUCTURES AND MACROS

The SPU switch over process, SPSNET_PROCESS, maintains information about the SPUs, the switchable sets, and the communication paths between the SPUs. This process is unnamed in a DPTREE trace; it always has a low PIN number.

The SPSNET_INFO macro displays information from the SPSNET_INFO record (type SPSNET_INFO_RECORD_TYPE). One of the fields in this record, SPSNET_MEMB, is an array of SPSNET_RECORD_TYPE. Each logical device (LDEV) has an element in this array. From these array elements the SPSNET_PROCESS determines which LDEVs are in each PBUS chain and uses this information to determine the communication paths for SPU to SPU communication. These array elements also contain other essential information for the SPSNET_PROCESS.

You must pass the PIN number of the SPSNET_PROCESS to the SPSNET_INFO macro. You can find this PIN number by looking at the stack traces of the low ordered PINs (begin with PIN 9) in the DPTREE until you find the one containing SPSNET_PROCESS.

DTC/TIO ENHANCEMENTS

SECTION 6

The Datacommunications and Terminal Controller (DTC/3000) is a LAN-based controller for asynchronous access to HP 3000 and non-HP systems, and for X.25 access to public and private X.25 packet-switching networks. For asynchronous access, DTC/3000 provides connections to terminals, PCs in terminal emulation mode, and serial printers.

In Release 3.0, the product supports duplex printing for the LaserJet 2000 and LaserJet IID. It increases the maximum number of supported asynchronous devices from 600 to 850. Also implemented in this release, is programmatic control of Typeahead Echo Mode through the FDEVICECONTROL intrinsic, which was introduced in Release 2.2.

PRODUCT ENHANCEMENTS

These product enhancements are described in the following paragraphs:

Duplex Printing

Duplex printing is the ability to print on both the front and back sides of the printer paper. Currently supported printers for duplex printing are the LaserJet 2000 and LaserJet IID. These two printers already support simplex (one sided) printing.

To utilize duplex printing on these supported printers, the user needs to configure printer type files with Duplex Printer Mode enabled through the TTUTIL.PUB.SYS utility. Then he assigns the profiles specifying the printer type files to the printer ldevs in NMMGR. This enables the Terminal I/O driver to send the appropriate escape commands to the printers.

The Duplex Printer Mode option is included in the Printer Control screen in TTUTIL as illustrated below:

[ ] Duplex printer (Y,N).

To indicate that a printer may be used for duplex printer, enter "Y" in the field.

In addition to assigning the appropriate printer profiles to the duplex printers, the user should also configure the printers properly, according to the printer reference manual.

After the configurations are completed, the customer can send the duplex printing commands (for example, Simplex/Duplex Printing Command Escape&l#S) to the printers, either by embedding the commands in the data of a file being printed, or by configuring the command as part of the printer initialization string within TTUTIL. Printers which do not support duplex printing will ignore duplex escape sequences sent to them, if they are misconfigured as duplex printers.

More Configured Devices

All Datacomm product configurations are done with the configuration utility NMMGR. The NMMGR configuration limit has been set to support 850 logons in Release 3.0, an increase from the 600 logons allowed in prior releases. Configuration errors caused by exceeding the limit are reported through

DTS/LINK Validation within NMMGR. In a PC-based DTC management environment, this number includes all nailed and non-nailed asynchronous devices.

For NMMGR screen information and configuration steps, refer to *Configuration Systems for Terminals, Printers, and Other Serial Devices (P/N 32022-90001).*

FDEVICECONTROL for Typeahead Echo Mode

The Typeahead Echo Mode option was implemented in Release 2.2 to specify the echo behavior, which permits either echoing input characters only once, or echoing characters twice. This option is offered on the Control Screen in TTUTIL.

Starting in Release 3.0, FDEVICECONTROL 192 with the control directive 63 is used to select single echoing or double echoing behavior for a particular DTC port. Refer to the *Asynchronous Serial Communications Programmer's Reference Manual* (P/N 32022-90002) for further details.

READER COMMENT SHEET

HP 3000 Commercial Systems

FIELD SUPPORT INTERNALS TRAINING
XL RELEASE 3.0 (Major Release B.30.00)

30216-90003 JANUARY 1991

We welcome your evaluation of this manual. Your comments and suggestions help us to improve our publications. Please explain your answers under Comments, below, and use additional pages if necessary.

Is this manual technically accurate? ☐ Yes ☐ No

Are the concepts and wording easy to understand? ☐ Yes ☐ No

Is the format of this manual convenient in size, arrangement, and readability? ☐ Yes ☐ No

Comments:

This form requires no postage stamp if mailed in the U.S. For locations outside the U.S., your local HP representative will ensure that your comments are forwarded.

FROM: Date _____

Name _____

Company _____

Address _____

_____

_____

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 1070 CUPERTINO, CALIFORNIA

POSTAGE WILL BE PAID BY ADDRESSEE

Publications Manager
Hewlett-Packard Company
Commercial Systems Division/44MX
19111 Pruneridge Avenue
Cupertino, California 95014

FOLD

FOLD

Part No. 30216-90003
Printed in U.S.A. 1/91
R3102

HEWLETT PACKARD