Managing MPE For Support

Bill Sutton
Hewlett-Packard
North American Response Center
Atlanta, Georgia

I. Introduction

Imagine the scene -- a desperate system manager at 2:00 in the morning. There is smoke pouring out of his Series 70 and the bit bucket threatens to catch fire. Our intrepid hero calls the Response Center for help. After ten minutes of frantic nail-biting, he receives the call As the soothing tones waft across the WATS line, what does our brave system manager hear?

Requests for information.

LOTS of information.

What is the problem (in detail)? What subsystems are involved? What are their versions? What version of MPE are you running? When was the last backup? When was the last UPDATE? When was the last COLDSTART? When was the last hardware change? When was the last software change? When was the last configuration change? Has this mix of applications ever worked before? Who was on the system and what were they running? What...when...how many...how long...<puff, puff>.

The questions can be frustrating at times, but all of them are necessary. To determine the cause of a problem, support personnel (whether hardware, software, or business) need to know the environment of the problem. There are many cases (ask other system managers at SIGBar and you'll hear lots of them) where the solution to the problem was in an obscure, cobwebby corner that never could have been searched without a complete set of answers to the endless questions above.

We can't get rid of the questions (of course, someday HP may develop an ESP interface option for PICS, but that's a future and we're not allowed to talk about it). The best we can do is to speed the process of getting those questions answered.

How can we speed that process? Unfortunately, until that ESP device gets to release 2 (Read-System-Mind-MIT), there is little we can do from the Response Center -- keeping a local database of customer configurations lags actual customer changes, dialling in can take as long as getting answers to the questions on the phone.

There is good news, however. There are techniques that can be used by the venerable and wise system manager to keep the information most often needed both up-to-date and readily accessible. Of course, first she needs to know what information to have and how to get it. Then comes the arcane rituals of job streams, gold book updates, low mumbles of the magic words "vuf" and "vufi," and all manner of trivia which up to now was passed down only in secret circles from grizzled system manager to wide-eyed innocent young assistant system manager.

No more! For at risk to my own life from the SPASMP (Society for the Preservation of Arcane System Management Procedures) I will reveal the forbidden secrets of:

Getting Fundamental Operating System information,
Getting HP Product information,
Getting Patch information,
Getting Error information,
Keeping System Activity information,
Keeping Application information,
Locating Important Tapes and Configurations,
and Tracking Backups and Coldloads.

II. Definitions

Before things start humming, it would be nice to make sure we are all talking about the same things when we talk about Support, System Management, and MPE. All of the above words or phrases are subjective; they have a slightly different meaning to each person who uses them.

Support is the most broadly-based term used here. Taken to extremes, every activity geared toward keeping a system up and running can be considered support -- including the people who run the vacuum cleaner in the operators' lounge. This is obviously too extreme for our purposes, unless we want to be here until August 1990. For use in this paper, Support will be defined as activity which keeps the hardware and Fundamental Operating System software functioning so that applications may be run.

System management is another sticky term. There are basically two types of system managers -- the administrative manager (in charge of purchasing and budgets) and the technical manager (in charge of bits and bytes). Sometimes both types are rolled into one. In our case, the System Management activities we are discussing are those performed by the system manager who spends his days with his sleeves rolled up and his fingers poised on the keyboard. In other words, these techniques are used to help discuss technical problems with technical support personnel.

Finally we get to MPE. Confusion reigns here, as we have MPE V (the old new operating system), MPE XL (the new new operating system), and MPE V/R

(the new old operating system). Suffice it to say that unless otherwise stated, the procedures here refer to MPE V systems and should also apply with minor changes to MPE V/R systems. There will be MPE XL examples scattered through this paper as needed.

All clear? Let's continue on to...

III. Vital Information

Vital information covers all those things you will almost always be asked to provide (system model, operating system version, etc.) Each piece of information is treated separately, with a few words on what it is, how to get it, what it is used for, and how to keep it up to date.

FOS VUF: Sounds like a French pastry, doesn't it? Actually, this is the Version Update Fix of the Fundamental Operating System. The Fundamental Operating System is all the software that comes with the system from HP at no extra charge -- this includes TurboImage, KSAM, Sort/Merge, the command interpreter, the file system, and many other modules. Version Update Fix is the version of the software and/or the operating system -- so called because it is in the form V.UU.FF (where Version = G, Update = A2, Fix = 04 (G.A2.04) for instance). The main VUF for the operating system is found on the SHOWME command:

```
USER: #S1751,BILL.SUTTON,INTEREX        (IN PROGRAM)
MPE VERSION: HP32033G.C2.02.  (BASE G.C2.02).
CURRENT: THU, MAY 12, 1988,  8:54 AM
LOGON:   THU, MAY 12, 1988,  8:36 AM
PROGRAM'S CPU SECS: 10          CONNECT MINUTES: 19
$STDIN LDEV: 24          $STDLIST LDEV: 24
```

The FOS VUF is the version you will be asked for most often. KNOW IT WELL. Often, a particular FOS VUF will have a name associated with it (G.02.00 is U-MIT, for instance). Always use the exact version number from the SHOWME command.

The scheme used for setting up the FOS VUF seems a bit strange until you get used to it. As a special added bonus, here is a quick rundown on each section.

The V (version) is the BASE RELEASE identifier. Versions of the OS with the same first letter will be based on the same tables layout, internal structures, etc. A change in the first letter signifies a major change in the OS direction. The most recent example would be from MPE IV (C.xx.xx) to MPE V/P (E.xx.xx) to MPE V (G.xx.xx).

The U (update) is the MIT identifier. A change in this number/letter combination signifies changes to particular modules of the OS which may

or may not affect the entire OS. An example of this would be the change from T-MIT (G.01.xx) to U-MIT (G.02.xx).

The F (fix) is the DELTA identifier. A change in this number signifies patches integrated into the release, new products, and/or new versions of old products.

What makes all this confusing is the fact that, in the past, patched versions and special versions of operating systems have been indicated by adding letters to the VUF (for instance, Q-Delta-2 was C.01.02 but the Q-Delta-2 Product Tape was C.B1.A2). Rest assured that this won't happen again until the next time it happens.

For MPE XL, things will be different. Each position in the VUF will represent a particular set of changes and the numbers will be sequential (0-9,A-Z). We might visualize an MPE XL VUF as M.CN.PN, where:

N: NL BUILD or POST-MR BUILD. This is a new creation of the Native Library either for special customer requirements or to integrate last-minute patches after the software has been released.

P: PATCH BUILD. This is the normal 'fixed in the next release' next release. It does not include new products or enhancements.

R: PRODUCTION RELEASE. This is a release which introduces new products or enhancements to current products. This type of release would also add support for new peripherals and software.

C: CORE RELEASE. This release contains major changes to the OS, TurboIMAGE, or datacomm. A core release will increase the functionality of the product changed.

M: MAJOR RELEASE. All bets are off, we rewrote a whole bunch. This position will always be a letter.

PRODUCT VUF: When your question or problem involves a specific product, it is the version of the product itself that matters. All HP software products (in other words, software that has a separate product number than FOS) have separate version numbers. Most third party software will also have a version associated with that package and/or individual modules as well. For HP's products there may be program versions, SL routine versions, XL routine versions, and overall versions.

For most products, the best way to get the version number is to run the program and look at the banner. It's usually a little tough to run an SL entry, so HP products with SL version numbers provide an interface you can use to get those numbers. Let's look at TurboIMAGE as an example:

```
:run query.pub.sys
```

```
HP32216C.00.05 QUERY/3000  THU, MAY 12, 1988,  8:57 AM
COPYRIGHT HEWLETT-PACKARD CO. 1976

>vers

    QUERY      C.00.05

IMAGE PROCEDURES:
    DBOPEN      C.00.30
    DBINFO      C.00.26
    DBCLOSE     C.00.28
    DBFIND      C.00.15
    DBGET       C.00.29
    DBUPDATE    C.00.15
    DBPUT       C.00.20
    DBDELETE    C.00.20
    DBLOCK      C.00.27
    DBUNLOCK    C.00.27
    BIMAGE      C.00.27
TURBOIMAGE PROGRAM FILES:
    DBSCHEMA.PUB.SYS    C.00.28
    DBSTORE.PUB.SYS     C.00.24
    DBRESTOR.PUB.SYS    C.00.00
    DBUNLOAD.PUB.SYS    C.00.00
    DBLOAD.PUB.SYS      C.00.00
    DBUTIL.PUB.SYS      C.00.17

>e

END OF PROGRAM
```

If your product versions are different in any way from those shipped on the SUBSYS tape, we will want to know! In almost all cases, a patch to an HP product or SL segment will change the version number.

For you datacomm folks, you may see a slightly different version format. In addition to Version Update Fix, you will also see an Internal Software Level (I) listed for each individual part of a module. This level is NOT compared with other parts of the module to determine module compatibility -- in other words, we do not put out an error or warning if the Internal Software Levels do not match. We do put out an error or warning if the Version, Update, or Fix levels do not match. For instance:

```
run nmmaint.pub.sys

NMS Maintenance Utility  32098-20010 A.01.03  ...

Subsystem version ID's:
```

```
Node Management Services 32098-20010 module versions:

SL procedure:  NMVERS00                    Version:  A0103023
SL procedure:  NMVERSCSL                   Version:  A0103024
SL procedure:  NMVERS01                    Version:  A0103000
SL procedure:  NMLOGSLVERS                 Version:  A0103010
SL procedure:  NMLOGDATAVERS               Version:  A0103014
SL procedure:  NMVERS04                    Version:  A0103007
SL procedure:  NMVERS05                    Version:  A0103000
SL procedure:  BFMVERS                     Version:  A0103002
Program file:  NMMAINT.PUB.SYS             Version:  A0103005
Program file:  NMFILE.PUB.SYS              Version:  A0103006
Program file:  NMLOGMON.PUB.SYS            Version:  A0103016
Program file:  NMDUMP.PUB.SYS              Version:  A0103043
Catalog file:  NMCAT.PUB.SYS               Version:  A0103004

Node Management Services 32098-20010 overall version = A.01.03
                         :
                         :
                         :
                         :
```

As you can see above, none of the Internal Software levels match for NMS. However, since the VUUFF portion matches, the overall VUF is consistant and therefore the software is consistent.

Appendix A contains a partial list of HP software products and how to get their version numbers.

HARDWARE: For hardware-oriented calls (as in the smoking system example given above), the most important piece of information you have is the serial number. A serial number is required whenever the Response Center has to place a call to your local CEO.

Often, contract information is also useful. If you are familiar with your support contract (hours, response times, etc.) it will help you make a more informed decision about whether or not to call the CE out at 3:00 AM for a down printer.

To combine the two, make a copy of your hardware support contract. Write the ldev number next to the serial number of each device listed on your contract. Keep this copy in your gold book.

PATCH INFORMATION: Suppose we have a segment (or a SOM for you XL fans) that doesn't have a version associated with it. How do we find out if a change has been made?

Each segment has a unique checksum associated with it (quite by accident in the beginning). Nowadays we store the checksum as it was at compile time in the patch area of each code segment. There is a program in telesup called CHECKSUM (amazing how they came up with that name) that will extract the segment checksum and compare it to one that the program calculates. There are three reasons for using this program:

1) IF the checksums match (stored and calculated) we have a compiled version of the segment. In other words, nobody went in and mucked about with SLPATCH and changed the segment (of course, they could have changed the checksum once they changed the segment but it's doubtful).

2) IF the checksums do not match, we either had a binary patch applied or something trashed the segment. Good time to do an update.

3) We can compare the checksum as given by CHECKSUM to the checksum of a patch for that segment. If they match, the patch has already been installed.

Another way to tell (not as accurate but easier to read) is to look in the MPEMIT33/HPSWINFO file. All install jobs for patches from IND and CSY now include a section which places the date/time/patchid in the MPEMIT33 or HPSWINFO file. This info is at the end of the file.

Why is this not as accurate? It happens when the install job is streamed, not when the patch is installed. If the install job completes successfully but nobody ever does a COLDSTART, the patch really hasn't been put in but the MPEMIT33/HPSWINFO file will reflect that it has. Take the information in these files as an indication but not as gospel truth.

What about MPE XL? Currently we have two kinds of patches available -- SOM (Software Object Module) replacement and binary. Currently the most prevalent are binary patches, as the NL is all one HUGE SOM and therefore is difficult to replace.

Binary patch tracking on XL is done through the patch program (SOMPATCH) itself. Before any information in the SOM can be changed, SOMPATCH requires the user to enter logging information. All changes are then logged (old values and new values) and may be listed out as follows:

```
(SOM 0),ID: 523563  Duane Souder  SR: 4700-523563
                                  Wed Apr 13 07:11:58 1988
; ----523563----Compiler Library----Duane Souder
; U_resume_execution calls P_close_files on
; ICS and page faulted. 10/30/87 KF. Apply to
; 9.91, 9.92, X.01. To START image only.
;
```

```
; Offset changed on the X.01 line. Change offset
; from 7
U_get_escapecode            + 54    e85f1f8d |  8000240
```

Since this information is part of the SOM, if it is present then the patch is present.

Unfortunately, there is no equivalent to CHECKSUM on the MPE XL machines at this time. For the most part it is unnecessary, as all modules of the OS and compiler librarys have associated version numbers. It has always been a useful tool for users to use in tracking versions, however, and perhaps someday a checksum program for XL will show up.

ERROR INFO: All too often, diagnosis of a problem is impossible because the error that occured is (or becomes) unknown. There are cases in which errors speed by at a rate too fast for the human eye to capture, but there are also cases in which the user simply fails to note the error message. A knowledge of how to collect 'lost' error messages can help a system manager solve many problems without a call to the response center.

In the system environment, error messages may come up on the user's screen or on the console. If the message comes up on the console and gets lost (scrolls off, console gets cleared, whatever) CONSOLE LOGGING can save the day. Logging all console messages will vastly increase the size of your system log files, but looking at the log (entry type 15 for MPE V, entry type 115 for MPE XL) will retrieve information you sometimes didn't know the system told you. As an aside, console log records are also useful for security monitoring, tape drive usage monitoring (how many tape requests did you get?), and terminal usage (what ldevs logged on between 8:00 and 5:00?). To keep disc space usage down, make it a standard practice to store/purge log files immediately after each full backup. You might want to leave a few days' worth on the system for easy accessability.

Messages that appear on the user's screen and then vanish are much more difficult to find. Prevention and/or duplication is a much better bet. Here are some methods for tracking, finding, and displaying hidden error messages:

1) Perform all error checking possible within your application. When you do this, you can store the error number and message in a disc file or send it to the console and so avoid losing error info. In many languages (COBOL, for instance), if you do not handle the errors explicitly the program will abort however the system wants.

2) If a VPLUS screen is causing problems identifying an error message, redirect the screen to another terminal. You can do this by issuing a file equation on the filename in the VOPENTERM intrinsic call (example: FILE A262X,NEW; DEV=ldev; ACC=INOUT).

Now any non-VPLUS IO will go to the terminal that the program is running on, while VPLUS IO will go to the other terminal. Note: both terminals must be from the same product family when you do this (as in 262x, 264x, etc.).

3) Some HP Products log their own errors (TDP FINAL, NS, etc.). Errors on these products can be found after the fact by looking in the product-specific log file. Some products must have logging enabled for this to work -- refer to the product manuals for details.

4) On MPE XL, the CI VARIABLES may be used to pass error numbers and messages back to the CI (to a UDC or command file, for instance).

5) ESC 0 on terminals with printers hooked up can log all pertinant information if the users are trained to use it by habit. LOG BOTTOM or COPY ALL can be used for non-VPLUS applications. Set a standard that users cannot report errors unless they have a hard copy of the screen with as much error information as possible.

6) Also on MPE XL, errors from MPE subsystems are logged to a PROCESS ERROR STACK. Currently, not all subsystems use this stack (which is available in DEBUG with the pm_errors macro). However, it is still a good resource for tracking chain-reaction type errors.

TIPS: Here are some tips that could be useful to help you communicate vital information quickly and clearly:

1) Track the version number, not the MIT name or "The version of QUERY that came out on UB-delta-4." Using the exact version number of the OS and products saves you lookup time when checking for known problems.

2) The file HPSWINFO contains the released version of every HP product on the current release. If a product has problems, check to see if its version matches that listed in HPSWINFO. For those on earlier releases of MPE (pre-UB-delta-4 [G.B2.04]), MPEMIT33 serves the same function. If the version number is LOWER than that in HPSWINFO or MPEMIT33, you may have a bad version.

3) When you are asked for product version information do NOT use the HPSWINFO version number! You may have had patches installed, be running an old version due to a problem with AUTOINSTALL, or have fallen victim to any number of quirks which may cause the version to be different than that released with the system.

4) Keep the versions of frequently-used products close at hand so that you won't have to run the products. See the job streams

provided with this paper for examples of various methods used to track versions at coldstart/update time.

5) For MPE XL users, remember that the Command Interpreter and the Operating System have different versions now. What you see when the CI starts up is the version number of the CI, NOT the OS.

6) Information on patches applied since OS or product installation is good to keep as well. Keep the patch ID number, who got you the patch (RC, SE, by mail from SDC, etc.), and when it was actually 'coldloaded' into the system.

III. HELPFUL INFORMATION

The information mentioned above is usually sufficient to solve most common problems. There are times, however, when a problem is so nasty and vicious that it continues to hide deep in the bowels of the system despite all efforts to dig it out. This means additional, environmental information may be needed.

Everyone has run into a problem that only comes up sometimes, even in the same function of the same program. The solution to this kind of error almost always involves the interaction of the program with other programs running on the system at the same time.

How can we find out what these environmental factors are? Can we get general system trend information which will give us an idea of what might be running at a given time of the day? Here are some methods to get that very information:

SYSTEM ACTIVITY: Some disc errors, memory pressure-related errors, and OS table-related errors occur only at certain levels of certain types of system activity. If you know the way your system tends to run at certain times of the day, investigation of the causes of these problems becomes much easier (and faster).

Of course, the easiest way of tracking overall levels of system activity is to let us do it for you. HPTREND gives hourly averages of CPU usage, IO activity, disc usage, and other resource activity. This type of indication will help define if an intermittent problem occurs only during periods of high <fill-in-the-blank> access.

Suppose you either don't have HPTREND (why not?) or your usage changes daily such that a monthly average doesn't help much. There are methods which, though they take a lot of time and manual calculation, can give the same figures.

The tools are OPT, SHOWJOB, the streams facility, a spare terminal, and (optionally) a spreadsheet program. The general method is as follows:

1) Get the following data from the batch report in OPT on an hourly (or more often) basis.

   -CPU : Busy, Pause Disc & Swap, MAM-ICS, Cache MAM-PROC STK, Cache MAM-ICS, Overhead.

   -Disc: All I/O, Reads, Writes, Control Ops.

   -Launches: Process Launches, Process Swap-Ins, Process Pre-empts.

2) Take this information and plug it into your spreadsheet on an hourly (or more often) schedule.

3) If you can plot from your spreadsheet, you can directly see how busy your system is in given time intervals. If not, you may wish to graph the results over time.

For a detailed look at how various programs or users affect your statistics, you may add the following processes to your setup:

1) Set the interval for the job above to no more than 5 minutes.

2) Stream a job which does a SHOWJOB with output appended to a permanent file, then streams itself 4.5 to 5 minutes later.

3) Run OPT from an unused terminal and get a program report logged every 5 minutes (300 seconds).

The combination of all this information tells you which users and/or which programs tend to have the greatest impact on the activity levels of your system. They also tell you what programs are run at the same time under normal circumstances -- meaning that if an intermittent problem shows up it will be easier to spot any deviation from normal processing.

This is excellent information to have at hand, but the amount of time involved in manually transferring and calculating the results of this activity is very high. It may be worth it in shops where usage fluctuates wildly from hour to hour or day to day, and of course the technique is very useful once an intermittent problem rears its ugly head.

Unfortunately, supported tools to provide all of these functions do not exist yet on MPE XL. HPTREND should be available Real Soon Now, and other performance related products should follow.

APPLICATION INFORMATION: Just as a large number of calls are solved using standard information as listed above, a large number of calls are solved by tracing the onset of the problem to a change in application code. How

do we find out whether changes were made, and how do we track who made them?

Unfortunately, HP does not rigidly force application version tracking. We do not require a version number as part of the object code, and we do not force an update to this version number with every compile. All of the methods available can best be expressed as voluntary and circumventable. For MPE V, we are very limited in our ability to imbed version information in object code. The most sophisticated method uses the Pascal $COPYRIGHT compiler option (also available in FORTRAN 77). Even though this places user written version information in the segment itself, there is no good way to print it out or access it.

One method of making sure changes have not been made is to always use the OLDDATE option on any RESTORE. This insures that the previous file create date (the compile date, for object code) is kept across restores. If a file's create date is recent, then, it follows that it is a new version of that particular program file.

In COBOLII, the DATE-COMPILED paragraph will bind a compile date into your object code. The WHEN-COMPILED reserved word may be used in your program code to reference or display this date.

On MPE XL, this issue is made a little easier. The program VERSION, which is in PUB.SYS and replaces PROGINFO and other unsupported utilities, accesses a part of the SOM known as the version area. In FORTRAN 77 and Pascal, a compiler directive, $VERSION, places user specified information in this area for VERSION to access. As an aside, the VERSION program also returns program capabilities, skeleton stack information, and maximum heap size, among others.

IV. TRACKING OS TAPES AND LISTINGS

If you have called the Response Center with a particularly solitary, nasty, brutish, and short problem there is a good chance that the solution path will come down to two options. One is to continue having the problem while a full-scale investigation continues. The other is to clean up the current version of MPE loaded on the system through an UPDATE or COLDSTART -- an action which will often clear the problem without a trace.

On a production system, clearing the problem is usually the choice. After all, the point of the system is to serve the users, not perform as a troubleshooting system for an obscure, one-time problem.

Too many times this solution fails because the caller could not locate a COLDLOAD TAPE from before the problem started occurring! In a worse scenario, the system is down, the only way to start it is from tape, and there is no tape to be found.

Anyone who has been in this situation has learned the hard way how important it is to keep track of OS tapes and their listings. It is equally important to toss old tapes and listings so that confusions (and sometimes career affecting actions) do not occur.

COLDLOAD TAPES: A COLDLOAD TAPE is defined as 'a tape created by SYSDUMP which includes system programs, drivers, the operating system, and @.PUB.SYS.' It is probably the most important tape you can have in regards to operating system integrity.

When you install your OS with AUTOINSTALL, the last thing it does is create a tape to load your system. THE INSTALL HAS NOT COMPLETED UNTIL YOU DO THIS! This tape is your ORIGINAL COLDLOAD TAPE.

From this point on, any time you make a configuration change, install a patch, or do any activity that requires creating a coldload tape, you will UPDATE from the IMMEDIATELY PREVIOUS COLDLOAD TAPE. This will insure that you have a completely clean version of the OS before you cut your new tape.

For example: you installed version X.YY.ZZ of MPE V last week. Now you need to add a printer. Before going into SYSDUMP to make the configuration change and cut the new coldload tape, you should UPDATE from the OCT (Original Coldload Tape). The new tape created by SYSDUMP will then have the exact same things (except for the new configuration) as the OCT, and any corruption which might have crept in during the week will not be propagated onto the new tape. This new tape now becomes the IPCT (Immediately Previous Coldload Tape).

From this the question arises: how many of these tapes should I keep? The answer, as usual, is that it depends on your site. If you do not make many configuration changes or install many patches, it certainly wouldn't hurt to keep all your coldload tapes for your current version of the OS. This gives you the option of backing up to any previous level of patches and/or configuration with just a single coldload.

If you have an active shop, however, the number of tapes could well be more than the amount of tape storage space you have. Your best bet would be to keep the OCT, the coldload tape just previous to the most recent change, and (of course) the coldload tape used to make the most recent change. You may want to keep more tapes than this if you change your system often (the more frequently changes are made, the more tapes back you may need to go to back out a problem). You should also store all patch files given to you before purging them from your system -- this way if you have to back out further than the number of tapes you kept you can still recreate your patch environment.

On MPE XL, the equivalent to the coldload tape is the SLT/STORE tape combination. Unlike MPE V, MPE XL does not store OS information and user files on the same tape set.

It is HIGHLY recommended that MPE XL users do a SYSGEN TAPE generation every time they do a full backup of the system. Also, for the store tape to be a true equivalent of the MPE V coldload user file portion, the store must be done with the DIRECTORY option.

BACKUPS: It only takes one RELOAD without a recent backup to discover that full and partial backups are necessary for the mental health of any good system manager. This is an example of Elephant Learning -- once you learn the hard way, you never forget.

Your operators should know where your most recent backups are. You should always keep at least the last full backup and all partial backups since on site. This is irrespective of security considerations, which may dictate restricted access to backup tapes and/or that all backups be kept offsite. Bear in mind that many system problems occur at night and gaining access to these tapes quickly could make the difference between a completed production run (albeit slightly late) and an open system management job. Seriously, security arrangements for tapes should include fast access for those authorized, not just safety for the tapes and privacy of the data.

OTHER CONFIGURATIONS: Many other products (datacomm, office products, etc.) require their own separate, more detailed configurations. Never forget that these configurations can be destroyed! Some sites maintain a separate tape which contains nothing but configuration files for their products (they use the indirect file feature of the STORE command to help in this). Keep in mind that often these products must NOT be in use (or even up) for the configuration file to be accessable.

V. IN CONCLUSION

The suggestions made in this paper are only a few of the ways to insure that important information about your system and its environment is always available. Every site has its own methods and standards for handling system information, but these methods should always include immediate access when needed as one of their goals.

If you follow the guidelines given above, you will find that the most tedious part of supporting your system -- gathering data -- will take less time and less effort. In addition, the data you have (especially in a down system environment) will be more accurate and therefore more useful to you, to HP, and ultimately to your users.

APPENDIX A
LIST OF PRODUCTS AND HOW TO GET THEIR VERSIONS

PRODUCT NAME	METHOD (MPE V)	METHOD (MPE XL)
ADCC SOFTWARE	:RUN TERMDSM.PUB.SYS	
ATP SOFTWARE	:RUN TERMDSM.PUB.SYS	
BASIC	:BASIC	
BASIC COMPILER	:BASICOMP	
BUSINESS BASIC	:BBASIC	:BBASIC
C		:CCXL (no prompt, enter :EOD)
COBOLII	:COBOLII	:COBOLII
COBOLII XL		:COB74XL or COB85XL
CS DOWNLOAD FILES	:RUN CSLIST.PUB.SYS	
CS INTRINSICS	:RUN CSLIST.PUB.SYS	
DS	:RUN DSLIST.PUB.SYS	
DS/X.25	:RUN DSLIST.PUB.SYS	
DTS		:RUN NMMAINT.PUB.SYS
DTC FIRMWARE		:DUI :RUN TERMDSM ST DT n
EDITOR	:EDITOR	:EDITOR
FORTRAN	:FORTRAN	
FORTRAN 77	:FTN	:FTN
FORTRAN 77 XL		:FTNXL
IMF/3000	:IMFMGR (MCHECK)	
IMAGE INTRINSICS	:RUN QUERY.PUB.SYS (VERS)	:RUN QUERY.PUB.SYS (VERS)
KSAM INTRINSICS	:RUN KSAMUTIL.PUB.SYS	:RUN KSAMUTIL.PUB.SYS
MRJE	:MRJE	
MTS	:RUN MPMON.PUB.SYS	
NRJE	:RUN NMMAINT.PUB.SYS	:RUN NMMAINT.PUB.SYS
NS	:RUN NMMAINT.PUB.SYS	:RUN NMMAINT.PUB.SYS
Pascal	:PASCAL	:PASCAL
Pascal XL		:PASXL
QUERY	:RUN QUERY.PUB.SYS	:RUN QUERY.PUB.SYS
RJE	:RJE	
SPL	:SPL	:SPL
SNA LINK	:RUN NMMAINT.PUB.SYS	:RUN NMMAINT.PUB.SYS
SNA/IMF	:RUN NMMAINT.PUB.SYS	:RUN NMMAINT.PUB.SYS
SORT/MERGE	:RUN SORT.PUB.SYS	:RUN SORT.PUB.SYS
TDP	:RUN TDP.PUB.SYS	:RUN TDP.PUB.SYS
VPLUS	:RUN FORMSPEC.PUB.SYS	:RUN FORMSPEC.PUB.SYS

```
!job coldinfo,manager.sys;hipri;outclass=lp,1,1
!comment
!comment          This job provides standard information
!comment          for use when placing support calls.
!comment
!comment          Version E.00.00 : MPE V/E
!comment
!comment
!comment
!comment      First, version and IO configuration info
!comment
!continue
!comment
!comment      In place of the SYSDUMP, you could use the
!comment      following:
!comment
!comment      !run sysinfo.prv.telesup
!comment      all
!comment      exit
!comment
!comment      In some cases this listing may be more useful since
!comment      it includes VM address information for each system
!comment      disc.
!comment
!sysdump $null
Y                    <<any changes?>>
                     <<system ID>>
                     <<memory size>>
Y                    <<IO Config Changes?>>
Y                    <<List IO devices?>>
Y                    <<List CS devices?>>
                     <<device defaults>>
                     <<highest DRT>>
                     <<ldev #>>
                     <<max open spoolfiles>>
                     <<list IO devices>>
                     <<list CS devices>>
                     <<terminal type changes>>
                     <<class changes>>
                     <<list IO devices>>
                     <<driver changes>>
                     <<IO configuration changes>>
Y                    <<System Table Changes?>>
                     <<CST>>
                     <<XCST>>
                     <<DST>>
```

```
                <<PCB>>
                <<IOQ>>
                <<DRQ>>
                <<TERMBUFF>>
                <<SYSBUFF>>
                <<SWAPT>>
                <<PRIMSG>>
                <<SECMSG>>
                <<SPEC RT>>
                <<ICS>>
                <<LST>>
                <<UCOP RQ>>
                <<TRL>>
                <<BKPT TABLE>>
                <<max user logging processes>>
                <<max users per logging process>>
Y               <<Misc config?>>
Y               <<List Global RINS>>
                <<delete global rins>>
                <<# of rins>>
                <<# of global rins>>
                <<# of seconds to logon>>
                <<max sessions>>
                <<max jobs>>
                <<default job CPU limit>>
                <<catalog changes>>
                <<softdump changes>>
Y               <<Logging Changes?>>
Y               <<List logging status?>>
                <<status changes>>
                <<logfile rec size>>
                <<log file size>>
Y               <<Disc allocation changes?>>
                <<max directory size>>
Y               <<List volume table?>>
                <<delete volume>>
                <<add volume>>
                <<list volume table>>
Y               <<Virtual memory changes?>>
Y               <<List virtual memory?>>
                <<vol name>>
                <<VM changes>>
                <<max # of spoolfile ksectors>>
                <<# sectors per spoolfile extent>>
                <<scheduling changes>>
Y               <<Segment limit changes?>>
                <<concurrent programs>>
                <<max code seg size>>
                <<max seg per process>>
                <<max stack>>
```

```
                    <<max xds size>>
                    <<max xds per process
                    <<standard stack size>>
                    <<system program changes>>
                    <<system SL changes>>
                    <<dump date>>
!comment
!comment      Now check for Datacomm things
!comment
!continue
!run nmmaint.pub.sys
!continue
!run dslist.pub.sys
!continue
!run cslist.pub.sys
Y
Y
N
!comment
!comment      Write information as to startup
!comment      into COLDDATE.PUB.SYS.
!comment
!file datetime,new;temp;rec=-80,,f,ascii;nocctl
!showjob ;*datetime
!setjcw cierror=0
!continue
!listf colddate;$null
!if cierror=0 then
!editor
text colddate.pub.sys
join datetime (#0/#0)
change 1,"System tape loaded on ",LAST
keep colddate.pub.sys,unn
exit
!else
!build colddate.pub.sys;rec=-80,,f,ascii
!editor
text colddate.pub.sys
join datetime (#0/#0)
change 1,"System tape loaded on ",LAST
keep colddate.pub.sys,unn
exit
!endif
!eoj
```

```
                    SAMPLE JOB STREAM -- MPE XL
!job coldinfo,manager.sys;hipri;outclass=lp,1,1
!comment
!comment          This job provides standard information
!comment          for use when placing support calls.
!comment
!comment          Version X.00.00 : MPE XL
!comment
!comment
!comment
!comment      First, version and IO configuration info
!comment
!continue
!sysgen
io
lc
ld
lp
lv
ex
lo
sh
ex
mi
sh
ex
sy
sh auto
sh dcc
ex
sh
ex
!comment
!comment      Now check for Datacomm things
!comment
!continue
!run nmmaint.pub.sys
!comment
!comment      Write information as to startup
!comment      into COLDDATE.PUB.SYS.
!comment
!file datetime,new;temp;rec=-80,,f,ascii;nocctl
!showjob ;*datetime
!setjcw cierror=0
!continue
!listf colddate;$null
!if cierror=0 then
!editor
text colddate.pub.sys
join datetime (#0/#0)
```

```
change 1,"System tape loaded on ",LAST
keep colddate.pub.sys,unn
exit
!else
!build colddate.pub.sys;rec=-80,,f,ascii
!editor
text colddate.pub.sys
join datetime (#0/#0)
change 1,"System tape loaded on ",LAST
keep colddate.pub.sys,unn
exit
!endif
!eoj
```

SAMPLE SYSSTART FILE

```
UPDATE
STREAMS 10
STREAM DATAJOB.PUB.SYS
****
COLDSTART
STREAMS 10
STREAM DATAJOB.PUB.SYS
****
```

```
** END OF FORMATTING **
TDP/3000     (A.04.01)  HP36578 Formatter
MON, JUN  6, 1988, 10:06 AM
NO ERRORS
INPUT  = EDITOR WORKFILE, TEXT FROM PAPER1
OUTPUT = *HP2680
```