



hp e3000
solution transition
advisor

**mpe commands
and networks**



white paper

table of contents



executive summary	1
overview	1
batch/jcl/utilities	3
batch processing on the e3000	3
streaming batch jobs	3
managing batch jobs	4
batch processing on other platforms	4
unix background processing	5
managing background processes	5
output of background processes	6
client/server applications on windows and unix	6
porting mpe applications with batch functionality to other platforms	7
tools: job schedulers	7
tools: translating mpe commands	8
tools: utilities	8
networking	9
data migration and retention	10
avoiding "magic weekends"	12
after the migration	13
appendix a: roman8 and latin1 character set comparisons	14
appendix b: intrinsics	18

executive summary

Because of HP's strategic decision to discontinue support of its HP e3000 platform, HP recognizes its customers are faced with making significant decisions on transitioning their HP e3000 solutions to other platforms. At this time, many questions arise on how to manage this process:

- What is the impact on affected business functions?
- What are the capital and human costs associated with transition efforts, both now and in the future?
- How can risk be mitigated?
- What is technically required for transitioning all or part of the HP e3000 solution?
- What are possible transition strategies?
- What are the appropriate Independent Software Vendors (ISV) migration tools or mix of tools to use in specific components of the HP e3000 software solution?

There are a total of six white papers in the "HP e3000 Solutions Transition Advisor" series. Each one addresses the subject of e3000 transitions from a different angle. The subjects of the six white papers are:

- Business Planning Guide
- Technical Planning Guide
- Compilers and User Interfaces
- TurboIMAGE and Databases
- KSAM and the MPE File System
- MPE Commands and Networks (this paper)

overview

This is the sixth of a series of white papers from Hewlett-Packard, providing HP e3000 customers with a transition process roadmap to assess, analyze, select and manage the appropriate transition strategy for their HP e3000 solutions. It is not intended to handle all HP e3000 transition issues, but a guide to:

- Batch Jobs — this white paper will present some guidelines for implementing batch portions of your applications on the recommended target platforms.
- MPE Commands — it will also present some alternatives for porting the portions of your applications that use MPE commands, either programmatically or from a colon prompt.
- Utilities — it will explore the various utilities used on the HP e3000, and identify corresponding utilities on the target platforms.
- Networking — it will identify the issues involved in porting networking functionality from the HP e3000 to the target platforms.
- Data Migration — it will identify the issues surrounding the migration of data files from HP e3000 to the target platforms

batch/jcl/ utilities

The HP e3000 is primarily an online transaction processing machine. Most users interact with the HP e3000 by logging onto the system using a desktop device such as a terminal or personal computer, and running application programs. These programs enable them to enter transactions which are then stored in databases and files.

Logging onto the system in this way creates an interactive **session**. By default, each session displays a “colon prompt” on that user’s desktop device. This permits the user to enter MPE commands, and have the results displayed on their screen. However, most HP e3000 installations are set up so that typical users never see a colon prompt. When a user logs onto the system, an application program is started on his behalf automatically - so that online users interact with the application, and never have to interact directly with the operating system. Typically, only certain specialized interactive users (e.g. the system administrator or the console operator) ever see an MPE colon prompt while logged on interactively. In addition, application programs can execute MPE commands using intrinsics (see Appendix B).

batch processing on the e3000

batch processing on the e3000

Most applications on the HP e3000 also have a batch component. The batch component is typically implemented in batch jobs, and handles the kind of work that is typically *not* done interactively. These kinds of tasks include the preparation of large reports or sorts, and other kinds of time-intensive processing.

- MPE defines a **batch job** as an independent execution entity managed by the operating system.
- A **batch file** is a collection of MPE commands coded in a file, which are used to specify which application programs will be executed by the batch job, using which files and/or databases.

streaming batch jobs

One can think of a batch job as a special kind of session. Whereas a session accepts commands and input from a desktop device, a batch job accepts it from a batch file. Whereas a session displays its output on the desktop device, the batch job sends it to an output spoolfile, typically for printing.

When porting HP e3000 applications to other platforms, it’s important to remember that any functionality that is provided via an MPE colon prompt must be made available on the target platform. In this white paper, we’ll examine ways that this can be done.

streaming batch jobs

An online user with a colon prompt can submit a batch file for execution by entering the :STREAM command. This is called “streaming” a job. Since most online users don’t interact with MPE’s colon prompt, batch jobs are more typically “streamed” by a console operator, or streamed programmatically by application programs.

The batch job is independent of the online user who streamed it. That is, if the online session that streamed a job logs off of the HP e3000, the batch job remains on the system, and executes independently of the online user.



managing batch jobs

batch processing on other platforms



managing batch jobs

The MPE operating environment provides the system administrator and console operator with a rich set of functionality for managing batch jobs. Much of this functionality can optionally be delegated to other users. For example, MPE's batch management environment includes the following features:

- **Limiting the number of batch jobs that can execute concurrently.** If more jobs are submitted than are allowed to execute concurrently, the system administrator can also control how jobs queue up for execution.
- **Controlling the execution priority of batch jobs.** This ensures that no batch job can monopolize system resources or cause performance problems for other sessions or jobs on the system.
- **Controlling the destination of the output of batch jobs.** This output may include reports and a log of the job's execution. It can be routed to any spooled printer, or kept in a file for retrieval and examination later.

batch processing on other platforms

Users of UNIX[®] systems and Windows[®] systems interact with the system somewhat differently from the way MPE users do. This reflects philosophical differences between the fundamental designs of the three operating systems.

- MPE was designed from the ground up to be used in commercial environments where end users are assumed to have little or no computer expertise.
 - Control of the system is reserved for a system administrator or console operator.
 - It is a server operating system only; there is no desktop version of MPE, and no built-in GUI.
 - Most MPE applications are *not* based on a client/server model. That is, all processes (whether batch or online) run on the server.
- UNIX was originally used in more technical environments, where the end users were more likely to have computer skills, and at least some familiarity with the command language (i.e. shell commands). This assumption has not carried over to commercial space, where users are more likely to be shielded from the underlying complexities of UNIX and the shell command language.
 - There are UNIX implementations for the desktop and for the server. The same command language applies to both.
 - The command language used with HP-UX is quite similar to the one used with Linux[®], or with other versions of UNIX.
 - Some UNIX applications are based on a client/server model, in which online processes run on a desktop machine, which interact with background processes that run on a server. But many use a more MPE-like model, in which all processes run on the server.
- Windows was originally designed for users of desktop computers.
 - Versions of Windows that are used with servers such as "Windows/NT Advanced Server" use the same graphical user interface as the desktop version.
 - Most Windows applications use a client/server model.



These differences can be seen in how end users are permitted to interact with the system. For example, on MPE, end users are permitted to stream batch jobs into the system. But the job of managing those jobs is reserved for a system administrator. MPE doesn't trust the end user with that task for fear that the user will monopolize system resources for his own benefit.

By contrast, UNIX puts much more of the management of the batch environment into the end user's hands, at least by default. This is because UNIX was originally designed around the principle that each end user has the perception that he has control over an entire system. Even if the user is logged onto a server with other users, the design philosophy behind the shell program is to make it appear that the user has a system all to himself.

A rich batch management environment like MPE's does not come bundled with HP-UX, Linux or Windows. Despite this difference, applications on these platforms are able to provide the kind of functionality that is typically handled in batch jobs on MPE. Sorts, Reports and other time-intensive tasks are typically handled using features of these platforms which allow work to be accomplished outside of an interactive session.

For example, on UNIX, users typically log onto the system interactively, much as they do on MPE¹. By default, UNIX users interact with the system using a **shell program**. There are a number of shell programs available for most UNIX systems, (including the Korn shell and the Bourne shell, among others). The command languages associated with these shell programs have a great deal in common — although each has features that make it unique.

When UNIX systems are used in commercial environments, the end users typically never see a prompt from a UNIX shell program. Just as we saw on the e3000, most commercial users interact with an application program which is run automatically when they logon. The application program acts as a shell program for the typical commercial UNIX user, and they never have to interact with the operating system directly. In commercial environments, the UNIX command language is used primarily in two ways:

- by specialized users (such as the system administrator)
- in non-interactive environments.

UNIX provides a variety of ways to execute a command or a file of commands (called a script) outside of an interactive session. If the user is running a shell program, (which is not the typical case in a commercial environment), he can use shell commands to initiate background processes on his behalf.

¹The command to logon to an e3000 interactively is "HELLO", while the command to logon to a UNIX machine interactively is "login". Both commands require that the user logging on identify himself with a userid and one or more passwords.

unix background processing



managing background processes



unix background processing

To run a script in the background, an online user would simply type in the script name (i.e., the name of the file containing the script), followed by an ampersand character (&).

- This will cause a new process, caused a **son process**, to be created. Commands in the script are then executed by the son process, freeing up the interactive user to do other things.
- When the son process finishes executing the commands in the script, it will terminate normally, and the online session will be notified of this fact.
- A son process is dependent upon the parent process that created it. That is, if you create a son process, and then log your interactive session off the system before the son process terminates normally, then the son process will be killed before it completes executing the commands in the script.

The use of son processes as described above represents the simplest analog to the MPE batch interface on UNIX systems. There are UNIX commands available with HP-UX and Linux that provide the online user with additional functionality for creating and controlling background processes.

- To make a background process *independent* of the session that started it, you can use the *nohup* command. This allows the script to continue executing even after the parent session logs off.
- The *at* and *batch* commands allow you to submit a script to be executed as an independent process in addition to specifying a start date and time.
- The *crontab* command allows you to set up scripts that will be executed according to a schedule. (*note: the at, batch and crontab commands all require that a cron daemon process be running, usually started at system startup.*)

Keep in mind, however, that the typical commercial UNIX user never sees a shell prompt. Instead, when he logs on, an application program is run automatically on his behalf. The user's interaction with the system is all done through the application program. In this case, UNIX application programs must include logic to perform the above kinds of tasks on behalf of the end users that are running them. If you are porting MPE applications to HP-UX or Linux, you may need to create UNIX-specific program logic to handle these kinds of functions.

managing background processes

Once you get a background process running, the UNIX operating system (by default at least) places much of the control of these processes in the hands of the end user. This is in line with the UNIX philosophy of making it look as though each end user of the system has exclusive control of his own system. It is different from the MPE philosophy of placing the management of batch jobs in the hands of a specialized user (the system administrator).

The UNIX command language assumes a fair amount of technical expertise on the part of the user. For example, shell programs provide the end user with a "process status" or *ps* command that can be used to display the processes on your system. Understanding the output of this command requires some fairly detailed knowledge of UNIX. For this reason, it's usually not practical to expect commercial end users to manage background processes using shell programs.

Good commercial UNIX applications provide users with the ability to create and manage background processes. But instead of forcing users to learn complex commands, the application provides this functionality as part of a simplified menu structure, in the context of the application's business logic.

output of background processes

output of background processes

Each MPE batch job that executes on an HP e3000 creates a listing that shows the commands that were executed and any error messages that may have been generated. This listing is usually sent to the system line printer spooler. This listing can be reviewed to ensure that the job executed properly. It can be always available as a log of what batch jobs have been run (successfully or not).

On UNIX no listings are generated automatically, at least not by default.

- Suppose a user who is interacting with a UNIX system via a shell program creates a background process.
- If the user wants to be able to examine the output of the process after it completes, he must redirect its standard output and standard error files (using the ">" character) to a file.
- This file can then be displayed while the process is executing, or after it terminates.

When porting e3000 applications to UNIX, it may be necessary to include logic in the application that save the standard output and standard error files, so that they can be displayed by the appropriate users to ensure that background processes completed successfully.

client/server applications on windows and unix

As is the case with UNIX, Windows operating systems do not include a robust subsystem for managing batch processing. Windows does provide programmatic interfaces that can be used to create son processes to do processing that is not appropriate for interactive users. Older versions of Windows used the MS-DOS command language for scripting purposes. With the transition to Itanium®, however, the MS-DOS command language will be a thing of the past; the scripting language used on Windows will have a lot in common with the shell command languages found on HP-UX and Linux.

Some UNIX applications and many Windows applications take advantage of a client/server model. In this model, the application is split between two machines: a client, and a server. The two machines are connected via a LAN, and the two parts of the application cooperate using the following guidelines.

- The end user interacts with a client program, which runs on a desktop machine which runs a client operating system such as Windows/XP Professional.
- The server runs an operating system that was tailored for use on a server, such as Microsoft's "Windows Server 2003", "Windows 2000 Advanced Server" or "Windows 2000 Datacenter Server."
- If the client program includes functionality that runs in the background, the programmer has the option of creating background processes that run on either the client or on the server.
- A Database management system such as SQL/Server will contain both client components and server components.



porting mpe applications

porting mpe applications with batch functionality to other platforms

As we've seen, the background processing environment that's available on UNIX and Windows is quite different from the batch processing environment that was designed and built into the MPE operating system.

When porting MPE applications to HP-UX, Linux or Windows, you should investigate how much of MPE's batch functionality was actually available to your end users, and decide how to continue making that functionality available after the application has been ported.

- If your end users had access to a colon prompt, then you may want to give them access to a shell program on the UNIX system. But if you do, this will require a significant amount of training because the shell command language can be quite complex, and in any case it is very different from the MPE commands that they have been using on the e3000.
- Alternatively, you may want to consider using an MPE emulation package. This provides all the functionality of a colon prompt on the target platform (HP-UX, Windows or Linux). MPE emulation packages are available from a variety of vendors, including Neartek and Bi-Tech.
- On the other hand, if your MPE application program provided your end users with the ability to stream jobs and display their output, then this functionality must be ported to the target platform along with the rest of your application logic.

MPE emulation packages such as those provided by Neartek or Bi-Tech provide the simplest method of porting applications from the HP e3000 to other platforms. These emulators make the MPE command language available on the target platform. Many of them also provide emulation for other MPE subsystems, such as VPLUS.

Regardless of whether your end users interacted with the e3000 from a colon prompt, or from an application program, the same commands that they used (or that your application programs used) on the e3000 can be used on the target platform with an MPE emulation package.

In the absence of an MPE emulator, an application that is being ported from MPE to UNIX must be enhanced to provide whatever batch functionality is necessary using the tools available on the target platform.

tools: job schedulers

The success of UNIX in the commercial marketplace has led to the development of a number of "job scheduler" products for HP-UX, Linux and Windows. Because these operating systems do not come with robust batch processing capabilities, software developers have taken it upon themselves to create them as software "add-ons" and make them available for sale. Many of these widely used packages provide batch processing environments for UNIX and Windows that are functionally similar to that which is available for MPE.

job schedulers

<u>product</u>	<u>comments/contacts</u>	<u>operating system</u>
Tivoli Workload Scheduler (TWS)	Formerly the Maestro Job Scheduler www.ibm.com	HP-UX, Windows, Linux
TIDAL Enterprise Scheduler	Formerly sys*ADMIRAL. Integrated into the Openview environment www.tidalsoft.com	HP-UX, Windows, Linux
Transport	Simulates MPE Job scheduler www.transport.bi-tech.com	HP-UX, Windows
ROC Task Services	New scheduler from the company that provides Maestro on the e3000 www.rocksoft.com	HP-UX, Windows, Linux



tools: job schedulers



tools: translating mpe commands

tools: translating mpe commands

MPE batch files are written using MPE CI commands (also called Job Control Language, or JCL). MPE JCL cannot be run on other systems without using an MPE emulator, such as the ones available from Neartek or Bi-Tech. But if your budget does not allow for the purchase of an MPE emulator, then there is a less expensive option.

MPE JCL can be usually be translated to a native command language on your target platform. To help with manual JCL conversion, HP is providing a command cross reference. This tool is available off the www.education.hp.com/curr-mpe-e3000.htm web site.

For more complex JCL there are translation tools, however the translation tools are only available as part of a consulting service.



jcl tools	product	comments/contacts	operating system
	AMXW	Supports MPE syntax JCL www.neartek.com	HP-UX, Windows
	ViaNova 3000	Translates MPE JCL to shell script www.ordina-denkart.com	HP-UX, Windows, Linux
	Intelligent Adapters	Translates MPE JCL to shell script www.transoft.com	HP-UX, Windows
	Transport	Supports MPE syntax JCL www.transport.bi-tech.com	HP-UX, Windows

tools: utilities

In addition to MPE commands and application programs, MPE batch files often make reference to utilities which are included with MPE. Three of the most commonly used utilities are:

1. FCOPY, which is used for copying files
2. SORT-MERGE which is used for sorting the data stored in a file and
3. EDIT, used for editing text files

Most commercial operating systems provide most, if not all, of the functionality of these utilities — but the commands and utilities, and the command syntax used to control the utilities is likely to be quite different from that used on the e3000.

For example, most of the functionality that FCOPY provides to MPE users is provided on HP-UX via a number of different shell commands, including *cp*, *awk* and *grep*.

The basic functionality of MPE's SORT-MERGE utility is provided on HP-UX using the *sort* utility. The bundled sort utility does not handle binary data so 3rd party sort utilities maybe required. Programmatic sorts are typically handled, not by a utility, but by the compilers, such as the COBOL compiler sold by AcuCorp. On the HP e3000, the MPE SORT-MERGE utility can be invoked programmatically using intrinsics. A list of these intrinsics can be found in Appendix B.



The editing of text files on the HP e3000 can be accomplished using the EDIT/3000 utility that is bundled with MPE/iX. This editor provides only the most basic text editing functionality, and many HP e3000 customers use more advanced text editors such as Qedit from Robelle. There are many text editors available (often for free) with UNIX systems; the most popular are the *vi*, *sed* and *emacs* utilities.

<u>utility</u>	<u>comments/contacts</u>	<u>operating system</u>
syncsort	Cross-platform sorting utility www.syncsort.com	HP-UX, Windows, Linux
suprtool	Popular data extraction and sorting utility from the HP e3000 www.robelle.com	HP-UX
qedit	Popular text editor from the HP e3000 www.robelle.com	HP-UX, Windows



networking

When the HP e3000 was originally designed in the 1970s, the internet was still in its infancy. Desktop computers did not yet exist, and so the networking model that was widely used at that time was very primitive by contemporary standards.

One of HP's early networking solutions for the HP e3000 was called NS/3000. NS (the acronym stood for networking services) was a collection of software and hardware products that collectively made it possible to connect multiple HP 3000s to one another via a network.

NS was not bundled with the HP e3000; it was a separate add-on product. If a customer purchased NS and installed it on all their e3000 systems, then users logged onto one HP 3000 in the network could access other HP 3000s via the network using an MPE command called :REMOTE.



The problem with NS was that it was very HP-centric. It was designed around the MPE and HP-UX operating systems, and assumed that your datacenter was made up primarily of HP machines. As computer networking progressed a non-proprietary collection of networking services was developed called ARPA (Advanced Research Projects Agency.) Although the NS network stack is the same in the transport layers as the ARPA services TCP-IP network stack. The services and applications that are available are different. For example network file coping with NS is done with DSCOPY and with ARPA it is done with ftp. The programmatic network access uses NetIPC calls with NS and ARPA uses BSD Sockets calls.

HP also had a product called "ARPA services" for the HP e3000 which enabled a TCP-IP stack for MPE/iX. This made it possible to connect an HP e3000 to a network using IP protocol. ARPA services included commonly used networking utilities such as ftp, but like NS, it was not bundled with every HP e3000 - it was an add-on product. Finally, in 1994, HP bundled ARPA services with MPE/iX, making it possible to use the HP e3000 in a corporate intranet without having to buy a lot of additional software.

Most HP e3000 applications pre-date the bundling of ARPA services into MPE/iX. Because networking was an "add-on" for so long in the HP e3000 world, comparatively few HP e3000 applications were designed to be dependent upon a network. This means that few HP e3000 applications have network dependencies in their source code.

If you're porting "network-aware" applications from MPE/iX to another platform, there are a couple of things to look out for:

- If the application is more than 10 years old, beware of dependencies upon the old NS networking subsystem. References to commands such as ":DSLIN" or the keyword ":REMOTE" are connected with the proprietary NS protocol. These references will need to be changed as part of the port to reference more open networking protocols.
- If the application is less than 10 years old, it may contain references to components of the ARPA Services package, such as the file transfer protocol (ftp) or the network file system (nfs). While these services are still widely available on UNIX, Linux or Windows, there may be far more efficient and intuitive ways of providing the same functionality.

There is a manual, "NetIPC to BSD Sockets and DSCOPY to FTP Migration Guide" (HP Part No. 98194-90043) that provides more detail network migration assistance. It can be found on www.docs.hp.com.

data migration and retention

data migration and retention

So far, we have been very focused on migrating application code from MPE/iX to your selected target platform. Once you have migrated your application code to the new platform you need to move your data, and undertake a testing program. Moving data between platforms can be complicated. There are a variety of migration tools available from third parties that can be used to move data between platforms, taking into account the complexities that can arise because of the different data-alignment and byte-ordering schemes used on different platforms.

In the case of ordinary text files, (such as program source files) a simple tool such as *tar* can be used on the HP e3000 to copy the files to an archive medium such as a magnetic tape. The POSIX standard for *tar* makes it possible to create a tar-tape on one platform (such as MPE/iX) and restore the files onto virtually any other POSIX-compliant platform such as HP-UX, Linux or Windows.

Once your HP e3000 is connected to your target platform via a network, you can use *ftp* to transfer files. There are products in development that will read an MPE/iX STORE tape on other operating systems and restore the MPE files onto the new OS.

When copying files from system to system, you need to be aware of architectural differences between platforms. Because of the HP e3000's roots as a 16-bit system, many binary fields in standard files are aligned on 16-bit boundaries on the e3000. Similarly, the byte-order used on different platforms can become an issue. Different processor architectures order the bytes of binary numbers differently.





You will have to consider the byte order in which multibyte numbers are stored, this is particularly important when you are writing those numbers to a file that is used for migrating data. The two orders are called “Little Endian” and “Big Endian”.

“Little Endian” means that the low-order byte of the number is stored in memory at the lowest address, and the high-order byte at the highest address. (That is, the “little end” comes first.)

For example, a 4 byte variable of type LongInt, Byte3 Byte2 Byte1 Byte0, will be arranged in memory as follows:

```
Base Address+0  Byte0
Base Address+1  Byte1
Base Address+2  Byte2
Base Address+3  Byte3
```

Intel® x86 processors use “Little Endian” byte order

“Big Endian” means that the high-order byte of the number is stored in memory at the lowest address, and the low-order byte at the highest address. (The big end comes first.) Our LongInt, would then be stored as:

```
Base Address+0  Byte3
Base Address+1  Byte2
Base Address+2  Byte1
Base Address+3  Byte0
```

PA-RISC processors use “Big Endian” byte order

Intel Itanium Processor Family-based Systems (IPF) can support both Big Endian and Little Endian storage schemes. So can PA-RISC processors, but MPE/iX and HP-UX use Big Endian exclusively. HP-UX will support Big Endian on IPF. However, Linux and Windows will support Little Endian on IPF.

Another factor in data migration is the character set. Most operating systems support the ASCII 7-bit character set called USASCII. This character set represents the US English upper and lower case alphabetic characters, numbers and some additional characters. To support non-US English characters an 8-bit character set that is a superset of USASCII was developed. At the time HP needed to support other language’s alphabets, there was not a standard 8-bit character set. So HP developed an 8-bit character set called ROMAN8. Since that time a new 8-bit character set was developed and has become the standard. This standard 8-bit character set is called ISO-8859-1 or LATIN1. Although HP-UX used to provide support for ROMAN8, today MPE is the only operating system that supports ROMAN8. When migrating data that contains the non-US-English characters, you may need to do some character translation. See appendix A for a ROMAN8 and LATIN1 comparison.

In order to compensate for these differences, a number of 3rd parties are offering migration tools specifically for copying data from one platform to another. It's especially important to use these tools when transferring binary data — such as that typically found in TurboIMAGE databases and KSAM files.

In the absence of a suite of migration tools, each TurboIMAGE dataset or KSAM file would have to be handled individually as follows:

- Extract the data from the TurboIMAGE dataset or KSAM file and put it into a standard (flat) file for transfer.
- Translate any binary data into ASCII for portability.
- Use FTP to transfer the flat file to the target machine.
- Now load the data into the new databases on the target platform.
- Some data transformation will be necessary as part of the extraction or loading process.

avoiding “magic weekends”

As is the case with any large project, planning is the key to success for large migration projects. Moving all the data and code to the target platform, and then switching production over in a single “magic weekend” is very rarely practical. There are a number of practices that have been used to ensure that software migrations are successful. Some of them are time consuming — but without them, you could wind up with a ported application that either doesn't work at all — or worse — works differently from the original application in ways that may not become apparent until days, weeks or even years after the porting project is “complete”.

Second to planning — testing it one of the most important activities in a migration project. There are two key aspects of your application that should be tested: testing the data, and testing the programs.

When migrating your data from platform to platform, it's always a good practice to test the validity of the migrated data, to ensure it has not become corrupted by the data migration process. This is especially important if your data files contain lots of binary information, which may be interpreted differently by different platforms. Some migration tools make it possible for your HP e3000 software running on MPE to access migrated data on your target platform. Using these tools, you can run your current application against the migrated data files. If your current application is able to do this successfully, it indicates that that data has been migrated successfully.

Similarly, when migrating code from e3000 to another platform, it's a good practice to test the migrated code against data on the e3000. There are migration tools that make it possible for software running on HP-UX, Linux or Windows to access files or databases on the HP e3000. Using these tools, you can run your migrated application against the original data. If the migrated application produces the same results as the original e3000 application, it is a strong indication that the code has been migrated successfully.

Similarly, **parallel testing** is a good technique for ensuring that application and data have been ported successfully. Once application and data have been ported to the target platform, they can be tested in parallel by running each day's production on both machines. If the data and application have been ported successfully, both applications should produce precisely the same results. To aid in parallel testing, products have been developed that can keep databases in sync. This allows for well-planned, thoroughly tested, phased migration projects.

avoiding
“magic
weekends”



after the migration



after the migration

After your e3000 applications have been migrated, tested and verified, you are ready to put the new application into production. Once this happens, you might be tempted to think that you will never again have any need for the e3000 or any of its associated subsystems. This is not always true. An important aspect of planning your migration project is to ensure that all your business needs are met, even after the migration is complete.

Many organizations occasionally need to access historical data. This may mean you'll need a means of accessing data from your HP e3000 applications, even after your migration project is complete, and the e3000 has been powered down for the last time.

One way to provide this access to historical data would be to maintain a collection of backup tapes (:STORE tapes) from your e3000. Of course, this assumes that you have a tool for accessing the data on :STORE tapes. One way to achieve this would be to keep a single old HP e3000 around for precisely this purpose. The key problem with this approach, of course, is that if the e3000 should break down anytime after 2010, (when HP will cease supporting it), you would lose your ability to access your historical data. Getting an obsolete e3000 repaired could be expensive, if it's possible at all.

Perhaps a better approach to solving this problems would be to take the old data that currently resides on :STORE tapes and move it to a more transportable medium.

- There are migration tools available today that can extract data from TurboIMAGE data bases and keep it in a logical format (for example order header and details kept together) in an archive file or in another database.
- There are products that can read a :STORE tape and restore the files on other operating systems.
- There are migration tools in development for accessing stored TurboIMAGE databases on :STORE tapes. These tools will be able to use the data from the :STORE tapes to create and load Eloquence databases.

	product	comments/contacts	operating system
data migration and archiving	Databridger	Extracts MPE data, moves and loads data on to other platforms www.taurus.com	HP-UX, Windows
	Bridgeware	Dynamically extracts MPE data, moves and loads data on to other platforms www.quest.com	HP-UX, Windows
	UDA Link	Moves/transforms MPE data to multiple platforms. www.mbfoster.com	HP-UX, Windows, Linux
restore tools	Rosetta Store	Can read Orbit and :STORE tapes www.imaxsoft.com	HP-UX, Windows, Linux
database synchronization	OpenTURBO	Syncs Image and Oracle databases www.imaxsoft.com	HP-UX, Windows, Linux
	IMAGO	Syncs Image and Eloquence databases www.omnisolutions.com	HP-UX, Windows, Linux

appendix a: roman8 and latin1 character set comparison



The 7-bit character set called USASCII is used by MPE (and most other operating systems) to represent the US English upper and lower case alphabetic characters, numbers and some additional characters. However, as computer applications have been enhanced to be more usable globally, the need has arisen to support non-US English characters. The term "Localization" has come to mean enhancing software to support various languages.

table A1. contains a list of MPE intrinsics that are used for Programming for Localization

<u>function</u>	<u>intrinsic</u>	<u>manual</u>
Retrieving information	ALMANAC NLGETLANG NLINFO	Native Language Programmer's Guide*
Handling characters	NLCOLLATE NLFINDSTR NLJUDGE NLKEYCOMPARE NLMATCH NLMATCHINIT NLREPCAR NLSCANMOVE NLSUBSTR NLSWITCHBUF NLTRANSLATE	Native Language Programmer's Guide*
Formatting time and date	NLCONVCLOCK NLCONVCUSTDATE NLFMTCALENDAR NLFMTCLOCK NLFMTCUSTDATE NLFMTDATE NLFMTLONGCAL	Native Language Programmer's Guide*
Formatting numbers	NLCONVNUM NLFMTNUM NLNUMSPEC	Native Language Programmer's Guide*
Using application message catalogs	CATCLOSE CATOPEN CATREAD NLAPPEND	Native Language Programmer's Guide*

* www.docs.hp.com/mpeix/onlinedocs/32650-90207/32650-90207.html

HP was a pioneer in supporting non-English alphabets. In order to support the development of localized software, an 8-bit character set that is a superset of USASCII was developed. At the time it provided this functionality on the HP e3000, there was not an industry standard for an 8-bit international character set. So HP developed an 8-bit character set of its own called ROMAN8. Since then, an alternate 8-bit character set called ISO-8859-1 or LATIN1 was developed and became an industry standard.

HP-UX used to provide support for ROMAN8. Today MPE is the only operating system that still supports HP's ROMAN8 character set. When migrating data that contains the non-US-English characters, you may need to do some character translation to overcome the differences between ROMAN8 (used on the e3000) and LATIN1 (used virtually everywhere else). The following Table (courtesy of ScreenJet) shows the correspondence between the ROMAN8 and LATIN1 character sets.



table A2. ROMAN8 and LATIN1 character sets

glyph	official iso 8859-1 (and unicode) name	roman8 hex	latin1 hex
	SPACE	20	20
!	EXCLAMATION MARK	21	21
"	QUOTATION MARK	22	22
#	NUMBER SIGN	23	23
\$	DOLLAR SIGN	24	24
%	PERCENT SIGN	25	25
&	AMPERSAND	26	26
'	APOSTROPHE	27	27
(LEFT PARENTHESIS	28	28
)	RIGHT PARENTHESIS	29	29
*	ASTERISK	2A	2A
+	PLUS SIGN	2B	2B
,	COMMA	2C	2C
-	HYPHEN, MINUS SIGN (HYPHEN-MINUS)	2D	2D
.	FULL STOP	2E	2E
/	SOLIDUS	2F	2F
0	DIGIT ZERO	30	30
1	DIGIT ONE	31	31
2	DIGIT TWO	32	32
3	DIGIT THREE	33	33
4	DIGIT FOUR	34	34
5	DIGIT FIVE	35	35
6	DIGIT SIX	36	36
7	DIGIT SEVEN	37	37
8	DIGIT EIGHT	38	38
9	DIGIT NINE	39	39
:	COLON	3A	3A
;	SEMICOLON	3B	3B
<	LESS-THAN SIGN	3C	3C
=	EQUALS SIGN	3D	3D
>	GREATER-THAN SIGN	3E	3E
?	QUESTION MARK	3F	3F
@	COMMERCIAL AT	40	40
A	CAPITAL LETTER A	41	41
B	CAPITAL LETTER B	42	42
C	CAPITAL LETTER C	43	43
D	CAPITAL LETTER D	44	44
E	CAPITAL LETTER E	45	45
F	CAPITAL LETTER F	46	46
G	CAPITAL LETTER G	47	47
H	CAPITAL LETTER H	48	48
I	CAPITAL LETTER I	49	49
J	CAPITAL LETTER J	4A	4A
K	CAPITAL LETTER K	4B	4B
L	CAPITAL LETTER L	4C	4C
M	CAPITAL LETTER M	4D	4D
N	CAPITAL LETTER N	4E	4E
O	CAPITAL LETTER O	4F	4F
P	CAPITAL LETTER P	50	50
Q	CAPITAL LETTER Q	51	51
R	CAPITAL LETTER R	52	52
S	CAPITAL LETTER S	53	53
T	CAPITAL LETTER T	54	54
U	CAPITAL LETTER U	55	55
V	CAPITAL LETTER V	56	56
W	CAPITAL LETTER W	57	57
X	CAPITAL LETTER X	58	58
Y	CAPITAL LETTER Y	59	59
Z	CAPITAL LETTER Z	5A	5A

glyph	official iso 8859-1 (and unicode) name	roman8 hex	latin1 hex
[LEFT SQUARE BRACKET	5B	5B
\	REVERSE SOLIDUS	5C	5C
]	RIGHT SQUARE BRACKET	5D	5D
^	CIRCUMFLEX ACCENT	5E	5E
_	LOW LINE	5F	5F
`	GRAVE ACCENT	60	60
a	SMALL LETTER a	61	61
b	SMALL LETTER b	62	62
c	SMALL LETTER c	63	63
d	SMALL LETTER d	64	64
e	SMALL LETTER e	65	65
f	SMALL LETTER f	66	66
g	SMALL LETTER g	67	67
h	SMALL LETTER h	68	68
i	SMALL LETTER i	69	69
j	SMALL LETTER j	6A	6A
k	SMALL LETTER k	6B	6B
l	SMALL LETTER l	6C	6C
m	SMALL LETTER m	6D	6D
n	SMALL LETTER n	6E	6E
o	SMALL LETTER o	6F	6F
p	SMALL LETTER p	70	70
q	SMALL LETTER q	71	71
r	SMALL LETTER r	72	72
s	SMALL LETTER s	73	73
t	SMALL LETTER t	74	74
u	SMALL LETTER u	75	75
v	SMALL LETTER v	76	76
w	SMALL LETTER w	77	77
x	SMALL LETTER x	78	78
y	SMALL LETTER y	79	79
z	SMALL LETTER z	7A	7A
{	LEFT CURLY BRACKET	7B	7B
	VERTICAL LINE	7C	7C
}	RIGHT CURLY BRACKET	7D	7D
~	TILDE	7E	7E
	NO-BREAK SPACE	A0	A0
À	CAPITAL LETTER A WITH GRAVE ACCENT	A1	C0
Á	CAPITAL LETTER A WITH CIRCUMFLEX ACCENT	A2	C2
Ê	CAPITAL LETTER E WITH GRAVE ACCENT	A3	C8
Ë	CAPITAL LETTER E WITH CIRCUMFLEX ACCENT	A4	CA
È	CAPITAL LETTER E WITH DIAERESIS	A5	CB
Î	CAPITAL LETTER I WITH CIRCUMFLEX ACCENT	A6	CE
Ï	CAPITAL LETTER I WITH DIAERESIS	A7	CF
´	ACUTE ACCENT	A8	B4
	MODIFIER LETTER GRAVE ACCENT	A9	

glyph	official iso 8859-1 (and unicode) name	roman8 hex	latin1 hex
	MODIFIER LETTER CIRCUMFLEX ACCENT	AA	
¨	DIAERESIS	AB	A8
˘	SMALL TILDE	AC	
Ù	CAPITAL LETTER U WITH GRAVE ACCENT	AD	D9
Û	CAPITAL LETTER U WITH CIRCUMFLEX ACCENT	AE	DB
	LIRA SIGN	AF	
-	MACRON	B0	AF
Ý	CAPITAL LETTER Y WITH ACUTE ACCENT	B1	DD
ÿ	SMALL LETTER y WITH ACUTE ACCENT	B2	FD
°	RING ABOVE, DEGREE SIGN (DEGREE SIGN)	B3	B0
Ç	CAPITAL LETTER C WITH CEDILLA	B4	C7
ç	SMALL LETTER c WITH CEDILLA	B5	E7
Ñ	CAPITAL LETTER N WITH TILDE	B6	D1
ñ	SMALL LETTER n WITH TILDE	B7	F1
¡	INVERTED EXCLAMATION MARK	B8	A1
¿	INVERTED QUESTION MARK	B9	BF
¤	CURRENCY SIGN	BA	A4
£	POUND SIGN	BB	A3
¥	YEN SIGN	BC	A5
§	PARAGRAPH SIGN, SECTION SIGN (SECTION SIGN)	BD	A7
f	LATIN SMALL LETTER F WITH HOOK	BE	
¢	CENT SIGN	BF	A2
â	SMALL LETTER a WITH CIRCUMFLEX ACCENT	C0	E2
ê	SMALL LETTER e WITH CIRCUMFLEX ACCENT	C1	EA
ô	SMALL LETTER o WITH CIRCUMFLEX ACCENT	C2	F4
û	SMALL LETTER u WITH CIRCUMFLEX ACCENT	C3	FB
á	SMALL LETTER a WITH ACUTE ACCENT	C4	E1
é	SMALL LETTER e WITH ACUTE ACCENT	C5	E9
ó	SMALL LETTER o WITH ACUTE ACCENT	C6	F3

glyph	official iso 8859-1 (and unicode) name	roman8 hex	latin1 hex
ú	SMALL LETTER u WITH ACUTE ACCENT	C7	FA
à	SMALL LETTER a WITH GRAVE ACCENT	C8	E0
è	SMALL LETTER e WITH GRAVE ACCENT	C9	E8
ò	SMALL LETTER o WITH GRAVE ACCENT	CA	F2
ù	SMALL LETTER u WITH GRAVE ACCENT	CB	F9
ä	SMALL LETTER a WITH DIAERESIS	CC	E4
ë	SMALL LETTER e WITH DIAERESIS	CD	EB
ö	SMALL LETTER o WITH DIAERESIS	CE	F6
ü	SMALL LETTER u WITH DIAERESIS	CF	FC
Å	CAPITAL LETTER A WITH RING ABOVE	D0	C5
î	SMALL LETTER i WITH CIRCUMFLEX ACCENT	D1	EE
Ø	CAPITAL LETTER O WITH OBLIQUE STROKE (LATIN CAPITAL LETTER O WITH STROKE)	D2	D8
Æ	CAPITAL DIPHTHONG A WITH E (LATIN CAPITAL LETTER AE)	D3	C6
å	SMALL LETTER a WITH RING ABOVE	D4	E5
í	SMALL LETTER i WITH ACUTE ACCENT	D5	ED
ø	SMALL LETTER o WITH OBLIQUE STROKE (LATIN SMALL LETTER O WITH STROKE)	D6	F8
æ	SMALL DIPHTHONG a WITH e (LATIN SMALL LETTER AE)	D7	E6
Ä	CAPITAL LETTER A WITH DIAERESIS	D8	C4
ï	SMALL LETTER i WITH GRAVE ACCENT	D9	EC
Ö	CAPITAL LETTER O WITH DIAERESIS	DA	D6
Ü	CAPITAL LETTER U WITH DIAERESIS	DB	DC
É	CAPITAL LETTER E WITH ACUTE ACCENT	DC	C9
ï	SMALL LETTER i WITH DIAERESIS	DD	EF
ß	SMALL GERMAN LETTER SHARP s (LATIN SMALL LETTER SHARP S)	DE	DF

glyph	official iso 8859-1 (and unicode) name	roman8 hex	latin1 hex
Ó	CAPITAL LETTER O WITH CIRCUMFLEX ACCENT	DF	D4
Á	CAPITAL LETTER A WITH ACUTE ACCENT	E0	C1
Ã	CAPITAL LETTER A WITH TILDE	E1	C3
ä	SMALL LETTER a WITH TILDE	E2	E3
Ð	CAPITAL ICELANDIC LETTER ETH (LATIN CAPITAL LETTER ETH)	E3	D0
ð	SMALL ICELANDIC LETTER ETH (LATIN SMALL LETTER ETH)	E4	F0
Í	CAPITAL LETTER I WITH ACUTE ACCENT	E5	CD
Ì	CAPITAL LETTER I WITH GRAVE ACCENT	E6	CC
Ó	CAPITAL LETTER O WITH ACUTE ACCENT	E7	D3
Ò	CAPITAL LETTER O WITH GRAVE ACCENT	E8	D2
Ö	CAPITAL LETTER O WITH TILDE	E9	D5
ö	SMALL LETTER o WITH TILDE	EA	F5
	CAPITAL LETTER S WITH CARON	EB	
	SMALL LETTER s WITH CARON	EC	
Ú	CAPITAL LETTER U WITH ACUTE ACCENT	ED	DA
	CAPITAL LETTER Y WITH DIAERESIS	EE	
ÿ	SMALL LETTER y WITH DIAERESIS	EF	FF
Þ	CAPITAL ICELANDIC LETTER THORN (LATIN CAPITAL LETTER THORN)	F0	DE
þ	SMALL ICELANDIC LETTER THORN (LATIN SMALL LETTER THORN)	F1	FE
.	MIDDLE DOT	F2	B7
μ	MICRO SIGN	F3	B5
¶	PILCROW SIGN	F4	B6
¾	VULGAR FRACTION THREE QUARTERS	F5	BE
—	EM DASH	F6	
¼	VULGAR FRACTION ONE QUARTER	F7	BC
½	VULGAR FRACTION ONE HALF	F8	BD
º	FEMININE ORDINAL INDICATOR	F9	AA
º	MASCULINE ORDINAL INDICATOR	FA	BA

glyph	official iso 8859-1 (and unicode) name	roman8 hex	latin1 hex
«	LEFT ANGLE	FB	AB
	QUOTATION MARK (LEFT-POINTING DOUBLE ANGLE QUOTATION MARK)		
	BLACK SQUARE	FC	
»	RIGHT ANGLE	FD	BB
	QUOTATION MARK (RIGHT-POINTING DOUBLE ANGLE QUOTATION MARK)		
±	PLUS-MINUS SIGN	FE	B1
‡	BROKEN BAR	n/a	A6
©	COPYRIGHT SIGN	n/a	A9
¬	NOT SIGN	n/a	AC
-	SOFT HYPHEN	n/a	AD
®	REGISTERED TRADE MARK SIGN (REGISTERED SIGN)	n/a	AE
²	SUPERSCRIP TWO	n/a	B2
³	SUPERSCRIP THREE	n/a	B3
¸	CEDILLA	n/a	B8
¹	SUPERSCRIP ONE	n/a	B9
×	MULTIPLICATION SIGN	n/a	D7
÷	DIVISION SIGN	n/a	F7

appendix b: intrinsic



table B1. shows a list of the intrinsic that are used for accessing features of the MPE/iX Command Interpreter

function	intrinsic	manual
Using commands programmatically	COMMAND HPCICOMMAND	Command Interpreter Access and Variables Programmer's Guide*
Controlling variables	HPCDELETEVAR HPCIGETVAR HPCIPUTVAR	Command Interpreter Access and Variables Programmer's Guide*
Controlling job control words (JCWs)	FINDJCW GETJCW PUTJCW SETJCW	Command Interpreter Access and Variables Programmer's Guide*
Identifying Parameter Input	MYCOMMAND SEARCH	Command Interpreter Access and Variables Programmer's Guide*

* www.docs.hp.com/mpeix/onlinedocs/32650-90493/32650-90493.html

table B2. contains a list of the intrinsic that can be used to invoke the MPE/iX sort/merge utility for the Sorting and Merging of Data

function	intrinsic	manual
Creating core merge routines (NM)	HPMERGEEND HPMERGEERRORMESS HPMERGEINIT HPMERGEOUTPUT	SORT-MERGE/XL Programmer's Guide*
Creating core merge routines (CM)	MERGEEND MERGEERRORMESS MERGEINIT MERGEOUTPUT	SORT-MERGE/XL Programmer's Guide*
Getting merge information (NM)	HPMERGESTAT HPMERGETITLE	SORT-MERGE/XL Programmer's Guide*
Getting merge information (CM)	MERGESTAT MERGETITLE	SORT-MERGE/XL Programmer's Guide*
Creating core sort routines (NM)	HPSORTEND HPSORTERRORMESS HPSORTINIT HPSORTINPUT HPSORTOUTPUT	SORT-MERGE/XL Programmer's Guide*
Creating core sort routines (CM)	SORTEND SORTERRORMESS SORTINIT SORTINPUT SORTOUTPUT	SORT-MERGE/XL Programmer's Guide*
Getting sort information (NM)	HPSORTSTAT HPSORTTITLE	SORT-MERGE/XL Programmer's Guide*
Getting sort information (CM)	SORTSTAT SORTTITLE	SORT-MERGE/XL Programmer's Guide*

* www.docs.hp.com/mpeix/onlinedocs/32650-90080/32650-90080.html

All brand and product names are trademarks or registered trademarks of their respective companies.

notice

The information contained in this document is subject to change without notice.

Hewlett-Packard makes no warranty of any kind with regard to this material, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Restricted Rights Legend

Use, duplication, or disclosure is subject to restrictions as set forth in contract subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software clause 52.227-FAR14.

Hewlett-Packard Company
3000 Hanover Street
Palo Alto, CA 94304, USA

© Copyright Hewlett-Packard Company 2006.
All rights reserved. Reproduction, adaptation or translation of this document is prohibited without prior written permission of Hewlett-Packard Company, except as allowed under the copyright laws

5981-3064EN rev. 3 (9/07)