JANUARY 1991

NOVELL® RESEARCH

NetWare and Microsoft Windows Integration

Jason Lamb
Senior Consultant
Systems Research Department

Abstract:

This Application Note will explain the integration of the Microsoft Windows environment with Novell's NetWare network operating system. Since Windows is a graphical user interface (GUI) that runs on top of MS-DOS, this Application Note will primarily deal with the integration issues for the NetWare workstation. We will also cover some network administration issues regarding Windows as well as some Windows theory and mechanics.

Contents

Acknowledgments

Put simply, this Application Note would not exist without the hard work and technical expertise of some very important individuals. First off, I want to extend my thanks to the person who brought this project to the Systems Research Department. He is:

Doug Knight Senior Systems Engineer

Doug continues to present seminars on Windows and NetWare Integration around the country, and it was he who provided all the initial, and much of the ongoing information and support for this project.

Another group of key individuals involved in this project, were the people in Novell, responsible for the development, marketing, and support, of the NetWare Windows Client software. They are:

Earle Wells Support Engineer

Karl Best Technical Writer, Windows

Tom Scribner Architect / Software Engineer, Windows

Danny Young Software Engineer, Windows

Andy Cox Product Manager, DOS/Windows Clients

Without this team there would be no issue with Windows and NetWare integration, because there would be no integration. Also, without Andy's constant drive this project could not have been finished in its time frame. Additionally, there were other engineers who, while not mentioned here, played an invaluable part in the production of this report.

Lastly, this Application Note was worked on up to the very last minute. This couldn't have been possible without the support of the people who run the Application Note Program.

Ron Lee Managing Editor

Bob Jones **Technical Writer / Desktop Publishing Specialist**

J. Warren Harding **Publisher / Manager Systems Research Department**

The only people left to thank are you. You as the reader, are the key to whether these, and other Novell information efforts, hit the mark. The only way we can know for sure, is if you let us know. Thanks.

Jason Lamb

Editor's Note: Feedback is encouraged to the author at FAX (801) 429-5511

Introduction

This Application Note will explain the integration of the Microsoft Windows environment with Novell's NetWare network operating system. Since Windows is a graphical user interface (GUI) that runs on top of MS-DOS, this Application Note will primarily deal with the integration issues for the NetWare workstation. We will also cover some network administration issues regarding Windows as well as some Windows theory and mechanics.

It probably comes as no surprise that the demand for this type of information is very great. Windows 3.0 has done more to excite the PC world than many products of recent memory. As might be understandable, the pace at which we tried to cover these issues was also great. As a result this Application Note has evolved out of a simple install and tips guide, into the monster you now hold in your hands. But don't fret if the size is a little daunting. We will go over each section of the Application Note below. From that you should be able to get a good idea what sections will be the most important to you.

We will begin by looking at the history of the Windows product line. This will cover from the very first shipped version, up to today's version 3.0. Following that we will look at two major pieces of the Windows environment, application support and memory management.

We will follow that with a look at something new for Windows, network support. This would be the first section to look at, if the concern is to know the installation issues first. Network services and network related files will be covered in depth in this section. Following that we will go through the process necessary for installing Windows on a NetWare file server, as well as the process for configuring Windows to run on NetWare workstations.

Next we will cover some startup considerations, as well as the important Windows and NetWare configuration files. Following that we will look at some run time considerations and finish up with some additional Windows LAN administration tips. The remainder of the document covers troubleshooting issues and appendices.

As this project evolved we found that our focus had also evolved. While we always felt that it was important to provide an installation guide and tips manual for the LAN administrator who needs to install software today, we also felt it was of equal importance that the LAN administrator gain some understanding of what makes Windows and NetWare Windows support, tick. It's our hope that this report can serve this two-fold purpose for people interested in the integration of Windows and NetWare.

Windows History

Microsoft first announced Windows as a GUI for MS-DOS computers, in 1983. However it would be two years later before Windows actually shipped to customers. In November of 1985 the first release of Microsoft Windows, version 1.01, became available. Windows 1.01 ran on a two floppy drive 8088 based PC with a minimum of 320KB of RAM. The most memorable feature of this version was the fact that the environment forced tiling of all the windows (rather than overlap them). This was done because it was thought that the automatic tiling would minimize the amount of work needing to be performed.

In 1987 version 1.01 was followed by Windows 2.0. Version 2.0 sported an enhanced look in order to

appear more like another Microsoft GUI announced that year, OS/2 Presentation Manager. Among its new features, version 2.0 allowed overlapping windows and introduced a new memory management scheme. Specifically version 2.0 added support for the Expanded Memory Specification (EMS), that had been developed by Lotus, Intel and Microsoft. This specification outlined a bank switching method of memory management that would increase available application memory under MS-DOS. Using this memory management scheme, Windows could provide applications with similar increased memory space.

With the introduction in 1988 of Windows 2.1, Windows itself was split into two separate products. The first product was Windows/286 and while this version could actually run on an 8088/6 CPU, Microsoft was now strongly recommending that the minimum processor be an 80286. Among its features, Windows/286 added support for the newly defined Extended Memory Specification (XMS), developed by Microsoft. XMS provides a standard method for allocating, using and releasing extended memory. Utilizing XMS and a modified version of Windows/286, this Windows environment could actually make use of the first 64KB RAM of extended memory for itself and Windows applications. This would actually be a prelude to the memory management architecture of Windows 3.0.

The second product was Windows/386. This version's main difference was multitasking support for Windows and non-Windows applications. Windows/386 utilized the 80386's virtual 8086 mode to run DOS applications in their own separate virtual machine.

Windows 3.0

The announcement of Windows 3.0 on May 22,1990 was one of the most celebrated roll outs of a computer software product ever. The fact that pre-release versions of Windows 3.0 had been widely distributed and available for some time to the developer and IS communities, did nothing to dampen the enthusiasm for the product when it was announced at simultaneous press announcements on both coasts.

In terms of the Windows product line the version number change from 2.1 to 3.0 was well justified. Windows 3.0 provides significant visual as well as structural changes from previous versions. Gone are the two separate products, Windows/286 and Windows/386. Windows 3.0 will run on either the 8088/6, 80286, 80386, or 80486 Intel processors, although Microsoft still recommends the minimum processor be an 80286.

Featured changes to Windows 3.0 are an enhanced look utilizing a proportional font, three-dimensional shading, color, and a new icon based desktop. Also, for the first time Windows now provides direct network support for certain networks. This includes the ability to connect to file servers and print servers from within the Windows environment. Novell NetWare support was written by Novell and is provided to Microsoft to be bundled in with the Windows 3.0 package. Figure 1 shows the new Windows 3.0 look.

Real Mode

One of the biggest changes with Windows 3.0 is that it can be run in either one of three different modes. These are Real, Standard and 386 Enhanced mode. Real mode is the recommended mode for running most Windows applications written prior to version 3.0. This mode requires 640KB of RAM and can run on any Intel 8088/6 processor and above. Real mode is the only Windows mode to utilize expanded memory for both Windows and Windows applications. Expanded memory conforming to the EMS 4.0 specification is required for this.

Standard Mode

The second mode is Standard mode and it incorporates among other things, a memory management technology designed to give access to the large protected mode memory address space of the 80286 processor or higher. However, it gives this access to Windows and Windows applications only. This memory management technique is called a DOS extender and it means that it is possible for Windows applications to be written which can access up to 16MB of RAM. This is what is meant when mention is made of Windows breaking the 640KB barrier.

Standard mode requires an 80286 processor or higher, as well as at least 256KB of extended RAM. This is RAM above the conventional RAM space of 640KB. Lastly, in order to run Windows in Standard mode you are required to load an Extended Memory Specification (XMS) memory manager, version 2.0 or higher.

386 Enhanced Mode

The third Windows 3.0 mode is the 386 Enhanced mode which as you might suspect, requires an 80386 processor or above. 386 Enhanced mode provides a similar DOS extender memory management technology as Standard mode. The difference is that Windows in 386 Enhanced mode provides virtual memory capability by using the paging features of the 80386 processor. By creating paging space on DOS disks for Windows to swap memory contents to, this allows Windows and Windows application to have access to more memory than is physically installed in the system. Windows in this mode actually provides virtual memory space up to four times the amount of physical memory installed. In its largest configuration this could result in Windows applications being able to access 64MB of RAM.

Additionally as in Windows/386, 386 Enhanced mode provides multitasking support by utilizing the virtual 8086 mode of the 80386 processor. Minimum requirements for this mode are 1 megabyte of extended memory above the conventional 640KB RAM, and an XMS memory manager, version 2.0 or higher.

The Windows Environment

Microsoft Windows is a GUI developed to run on top of MS-DOS. However, Windows will typically take over many of the standard operating system chores from MS-DOS. It is for this reason that Windows is described as an operating environment. Windows, in fact, handles much of the application support that is usually associated with an operating system, such as user input, graphics output to displays and printers, and memory management. It does however, use MS-DOS for some services like file I/O, and other disk operations.

There are two areas of the Windows environment that are important to understand, in terms of Windows and network support, and in terms of understanding the Windows environment in general. These areas are Windows applications support, and Windows memory management.

Windows Applications

Within the Windows environment there is support for two different types of applications. The first is the Windows specific application. This type of program is written to use all the system services provided by Windows, and consequently cannot run without loading at least a minimal version of Windows first.

Because the Windows environment provides new or changed access to system resources, application programmers have to develop applications that directly call these resources if they are to take advantage of the Windows environment. This has the benefit of releasing the application programmer from having to develop routines that handle system resources such as the screen output, printer output, and user interface. The programmer can just make use of the standard Windows resources to handle these tasks.

Non-Windows Applications

This second type of application supported by the Windows environment, is the DOS application not written to run under Windows. These types of applications are sometimes referred to as standard applications (in versions of Windows prior to 3.0), non-Windows applications (with the introduction of Windows 3.0), or simply DOS applications.

Since there exists a large amount of standard DOS applications, Microsoft has equipped Windows (since early versions), with the ability to run these standard MS-DOS applications without exiting the Windows environment. Version 3.0 provides support for running these non-Windows applications from within Windows, in all three modes Real, Standard and 386 Enhanced.

Windows programmers, since early versions of Windows, informally provided another classification for non-Windows applications. These programmers would refer to non-Windows applications as old apps. They further split old apps into both good old apps and "bad old apps". The classifications of "good" and "bad" were not in regards to the perceived quality of the application, but rather in how this type of application would behave while running under Windows. The way in which the application was written to access the PC, directly determines much of its ability to run from within Windows.

"Good old apps" were non-Windows applications that were character based and written to use all DOS and BIOS services, as opposed to directly accessing the hardware. The "good" in "good old apps" is applied only for the reason that Windows could successfully trap many of the system calls these types of applications would make, and translate them into appropriate Windows calls. To the user prior to Windows 3.0, these would be the types of non-Windows applications that could be run from inside of a window on the Windows desktop.

"Bad old apps" were non-Windows applications that were typically graphical based and made calls directly to the hardware. Since Windows could not trap all the system calls these types of applications would make, these would typically have to be run in, what was known prior to version 3.0 as, full screen mode. This would allow the application to access system resources, in particular the screen, more directly.

Windows/286 and Windows/386 could even go as far, as to swap Windows applications and most of the Windows environment out of memory to disk, in order to let the "bad old app" have full access to system resources. In the Windows/386 product this was referred to as running an application in exclusive mode.

Most prior problems with running non-Windows applications came from these "bad old apps". Even when running these types of applications in exclusive mode it was possible to issue system calls that would conflict with or confuse Windows.

Windows 3.0 has refined previous, as well as added further, support for running non-Windows applications. This added support is different based upon which mode of Windows you are running. Real and Standard modes support non-Windows applications by utilizing task switching for the separate DOS sessions and the Windows session. 386 Enhanced mode supports non-Windows applications by creating virtual machines for each separate DOS session and the Windows session.

Real and Standard Mode Support

In Real and Standard mode running non-Windows applications can only be done in what was previously referred to as exclusive mode. Windows and Windows applications are suspended while the system is given over to the non-Windows session. This means that you cannot run these non-Windows applications inside of a Window while in Real and Standard modes.

In Real and Standard mode, Windows will perform task switching. This means that when a non-Windows application is run, either from a DOS prompt called from within Windows, or when the program is called directly from within Windows (to Windows there is no difference), Windows will create a session for it and allocate a block of RAM for its use. In order to allocate this block of memory, Windows will switch certain items out of the lower 640KB of RAM, or conventional RAM, to either extended memory or to disk. Windows will then suspend itself and any running applications, in order to let the called non-Windows application run.

When another non-Windows application is run, or when the Windows session is called up, you will see a message saying Switching while Windows again switches a portion of the current contents of the lower 640KB of RAM out to either extended memory or disk.

Note that the entire contents of the lower 640KB of RAM is not swapped out when this task switching is performed. Windows maintains what is referred to as global memory which is not switched out of the lower 640KB of RAM, and which contains system information such as the COMMAND.COM processor, Terminate and Stay Resident (TSR) programs, and network or other drivers. This global memory is used as system resources for each successive switched session. Memory which is switched out is used only by its specific

application and is referred to as local memory.

Operationally this provides a single global memory component that is never swapped out of the lower 640KB of RAM, and separate switchable local memory components (one for each session running) that are swapped in and out of the lower 640KB of RAM. Figure 2 shows how the Standard mode, task switching process works.

386 Enhanced Mode Support

Windows 386 Enhanced mode provides a different method of running non-Windows applications. As with Windows/386, 386 Enhanced mode utilizes the virtual 8086 mode of the 80386 chip to create and maintain separate virtual machines, for both Windows sessions, and non-Windows application sessions. Using this mode of the 80386 allows Windows to isolate each session from each other, and to virtualize access to system resources among all sessions.

The technique of virtualization is not new to computer technology. It basically involves convincing the calling application that it has direct access to the system resources, whether it does at that moment, or not. A simple analogy might be the phone system. With all of the various phone systems installed across the country, a user never knows over which wires (or whether in fact it's wires at all), or through which route their particular phone call might take. They can assume that it is a certain connection through a certain route, and as far as the phone call is concerned it doesn't matter whether they're right or not. The phone call operates in exactly the same way regardless. It's the phone company, or rather the phone company's computers, that retain responsibility for providing the actual connection and route.

Using this technique Windows virtual device drivers allow access to the system resources, to be shared among all virtual machines. Virtual machine support under 386 Enhanced mode is a combination of software, as well as hardware management of the virtualization. This combination is much more powerful than any similar software only based approach.

As a result, these virtual machines allow Windows to successfully trap system calls from non-Windows applications, since these calls must be given to the virtual device driver first. It is for this reason that under 386 Enhanced mode, unlike Standard mode, non-Windows application sessions can be run in a few different ways. They can be run inside of a window on the desktop, in full screen mode, multitasked in the background, or in exclusive mode.

Like Windows in Standard mode, 386 Enhanced mode Windows maintains a global memory segment for system information like the COMMAND.COM processor, drivers and other TSRs. This global memory is memory located in the first virtual machine, which is the Windows session. This virtual machine is sometimes referred to as the System VM (Virtual Machine) or as VM0. However unlike Standard mode, in 386 Enhanced mode when a new virtual machine is created, the global memory of VM0 is simply remapped as the lower memory of the new virtual machine. This is done using the virtual memory addressing capability of the 80386. Figure 3 represents the 386 Enhanced mode's virtual machine mechanics.

PIF Files

Even with the enhanced non-Windows application support available in Windows 3.0 it is sometimes advisable to guarantee the amount and type of system resources for each non-Windows application session. Windows allows a user to configure this through the use of Program Information Files or PIF files. These files are covered in depth in the Microsoft Windows User's Guide pages 440-490. All settings in both standard and 386 Enhanced mode are explained in this section of the manual. Figure 4 shows the PIF settings available in 386 Enhanced mode.

Figure 5 shows the settings available through PIF files in Windows Standard mode.

Again, for a complete discussion on these settings please consult the Microsoft Windows User's Guide pages 440-490.

Windows Memory Management

In this next section, rather than explain various PC memory management techniques, such as expanded memory, extended memory, and DOS extenders, we will instead concentrate on explaining the way in which PC memory is managed among Windows and non-Windows applications, in all three Windows modes.

Real Mode

As mentioned in the opening introduction, Real mode Windows requires an 8088/6 CPU or higher and at least 640KB of RAM. Also, Real mode Windows can utilize EMS 4.0 expanded memory. This means that Windows and Windows applications can have access to more than 640KB of RAM through the use of expanded memory only. Non-Windows applications also have access to more than 640KB of RAM through the use of the same expanded memory.

In order for this to happen two things must exist. The first, is EMS 4.0 software and hardware needs to be installed on the system. The second, is that the Windows application, (as does the non-Windows application), needs to be written to use expanded memory. This is more typical of Windows applications prior to version 3.0, since this was the way in which larger memory address space was available in older versions of Windows. (The first XMS support only provided Windows and Windows applications with an additional 64KB of usable memory).

Version 3.0 Windows applications would not utilize expanded memory because Windows 3.0 introduces a new memory management scheme that is far more powerful than expanded memory. Most informal recommendations suggest only using Windows Real mode, and expanded memory Windows applications, as an interim step before moving up to more powerful modes of Windows.

Standard Mode

Standard mode introduces Windows applications to the larger protected mode address space of the 80286 chip. In order to run Windows in Standard mode you must have an 80286 CPU or higher, at least 256KB of extended RAM and an XMS memory manager, version 2.0 or higher.

Windows Applications

Utilizing the DOS extender technology described in the opening section, Standard mode allows Windows and Windows applications to access up to 16MB of RAM. If you refer back to you will see that Windows maintains a global memory component and switches out various local memory components based upon what is called to run.

It should be clear that even though the figure depicts all local memory components to be the same size, in operation they are not. Due to the DOS extender technology utilized as part of Windows memory management, the Windows session can be much larger than standard non-Windows sessions. The only thing that is required to take advantage of this is to run Windows in Standard mode and to run Windows applications written to run under Windows 3.0.

Non-Windows Applications

As previously mentioned non-Windows applications running under Windows Standard mode face, task switching when they are called upon to run. They can only be run in exclusive mode and they have access to a 640KB window of memory to run in. While it might be nice to think that this window of memory is reserved for the application alone, it is not. This 640KB window is comprised of both the global, and local memory components, spoken of earlier. This is the manner under which most non-Windows applications will run in Standard mode.

One small caveat here, is that non-Windows applications are only usually limited to the 640KB window, mentioned above. If the non-Windows application is written to use the same DOS extender technology as

Windows, then this application can simultaneously have access to the same memory space as Windows and the Windows applications. Additionally, through the use of PIF files, you can control the amount of memory given over to the DOS extender compliant, non-Windows application.

Finally even though Standard mode Windows does not use expanded memory for either Windows or Windows applications, a non-Windows application can have access to expanded memory. The expanded memory must simply be installed on the system (and not conflict with any Windows memory managers) prior to loading Windows, and it will be available to the application.

386 Enhanced Mode

386 Enhanced mode Windows requires an 80386 CPU or higher, 1MB of extended RAM, and an XMS memory manager, version 2.0 or higher. 386 Enhanced mode provides all the same memory management techniques available in Standard mode, with a few notable additions. One addition is the previously mentioned virtual memory capability of 386 Enhanced mode. In this mode, a Windows applications can access up to 64MB of RAM. A second addition is that 386 Enhanced mode also provides EMS 4.0 expanded memory for non-Windows applications. This is done all without loading any EMS memory managers.

DOS Protected Mode Interface (DPMI)

While DOS extender technology is not new by PC standards, Microsoft's implementation of the one in Windows is new. The Windows DOS extender technology was announced with the introduction of Windows 3.0, and it has recently been published as a specification from Intel. It is called the DOS Protected Mode Interface (DPMI). As with Standard mode, DPMI compliant non-Windows applications can have simultaneous access to the large virtual memory address space of Windows in 386 Enhanced mode. You can also limit the amount of DPMI memory available to an application by using a 386 Enhanced mode PIF file.

The only problem with a new DOS extender is that it can easily conflict with other DOS extenders. The most popular DOS extender to date is just such an example of a conflicting one. The older Virtual Control Program Interface (VCPI) is the most widely implemented DOS extender and it does conflict with DPMI. Because of this conflict, as of version 3.0, Windows in 386 Enhanced mode will not allow a VCPI application to run in a non-Windows session.

Windows and Networks

Improved network support is one of the most significant enhancements in Windows 3.0. Prior versions of Windows were mostly ignorant of network operations, either treating such operations as server disk access as if it were local drive access, or by not supporting certain network operations at all. With version 3.0 much of this has changed.

Windows Network Services

Microsoft developed Windows 3.0 to support various types of networks through the use of Windows network drivers. Certain types of network operations like mapping drives and connecting to servers and print queues, can be done through the use of Windows network drivers, and as such, these operations can be done from within the Windows desktop.

The network services provided to Windows users through the NetWare Windows driver, allows Windows users to attach and detach to any NetWare server, although you cannot login or logout of NetWare servers. You can also add, change, and delete, drive maps as well as connect, and disconnect, from any NetWare print server and queue. Lastly, NetWare support includes the ability to receive network messages while within the Windows desktop.

Along with the ability to perform certain types of network operations from within Windows, you can also install Windows exclusively for network workstations, including diskless workstations. This involves keeping

a shared directory of all Windows files on the server and utilizing a new SETUP option to install only the necessary unique Windows user files, into separate user directories.

Windows/NetWare Files

Prior to installing Windows on a NetWare network it is necessary to have all the proper files. These include files from the Windows distribution diskettes, and files from Novell. The Novell supplied files are available from a variety of Novell sources including the CompuServe Information Service - Novell Support Forum, NetWire. The following is the list of the necessary Windows/NetWare files and a description of their function.

Novell-Supplied Files

In order to determine if you have the necessary Novell-supplied files and versions, consult Appendix A which contains a complete list of necessary files listing the filename, size, and date. Appendix A also lists where these files can be obtained from NetWire. These new NetWare shell and utilities software are not exclusively for the operation of Windows, but these are the only NetWare shell version and requisite utilities that will properly operate with Windows 3.0. The latest version of the NetWare shell is v3.01e.

Special Notes - Novell Files

Special care must be made in regards to the BINDFIX utility. Running prior versions of BINDFIX with the newer NetWare shells can create problems. Before using BINDFIX with the new NetWare shells, please read Appendix B, which includes the Novell Technical bulletins 255 and 256. These bulletins cover BINDFIX problems, resolutions and the latest program version listing.

It is also recommended that if you use TBMI.COM you should remove VIPX.386 (which is explained in the Windows Supplied Files section below) from your system. In order to do this you must edit the SYSTEM.INI file (explained in the Windows/NetWare Configuration Files section later in this document), to remove the automatic loading of VIPX.386.

Windows-Supplied Files

Except where noted, the following files ship with the Windows 3.0 distribution diskettes.

Installation

Prior to installing any software on a network server, it is important that all licensing issues be handled, as required by the software manufacturer. Please consult Microsoft licensing material, to determine the necessary procedure for correct licensing of Windows on a network.

The bulk of this installation section will cover Windows installation for shared server access, with just minimal workstation Windows user files. If it your desire to install full copies of Windows locally for each NetWare workstation, you can skip the following sections until Startup Considerations.

The basic steps necessary to installing Windows on a NetWare network are simple.

The remainder of this section will go into more depth on each one of these steps, as well as explain certain startup and runtime considerations for running Windows on a NetWare network.

Setup NetWare Workstations and Servers

Prior to installing Windows on the network, it is important that all workstations have the correct NetWare shells. This includes correct versions of IPX.COM and either of the three shell software components, NETx.COM, EMSNETx.EXE, or XMSNETx.EXE. As stated previously, it is recommended that if there is a choice between using one of the two upper memory shells, EMS or XMS, it is recommended to use the XMSNET shell. This is because Windows utilizes XMS, and not EMS memory, in Standard and 386

Enhanced mode.

After the NetWare workstations have had the proper shells loaded on them you need to have all NetWare servers updated with the proper version of NetWare utilities. Take special care with the BINDFIX utility. Prior to running BINDFIX with the new NetWare shells, please read the Technical Bulletins located in Appendix B.

If you are unsure whether you have the correct files or not, Appendix A contains a complete list of the necessary NetWare utility and shell files, listing the filename, size, and date. Appendix A also describes where on NetWire these files can be obtained.

After all workstations and servers have been updated the next step is to configure the Windows software on the server.

Setup Windows Shared Files

The first step in doing this is to create a Windows shared directory on the server. It is from this directory that each users' specific version of Windows will be built. In order to do this you only need to use the Microsoft supplied utility EXPAND.EXE to uncompress the compressed Windows files and copy them to a single directory on the server.

The batch file documented in the Microsoft Windows User's Guide on page 553, can accomplish this is as follows:

After copying the program EXPAND.EXE and the newly created batch file EXPALL.BAT to the F:\WINDOWS directory, you can then insert the first Windows diskette in drive A and run the EXPALL batch file.

Simply repeat this step for all the supplied Windows diskettes. Once all files have been expanded you can rename all of the Windows device drivers from *.SY$ to *.SYS if you wish. (Although the SETUP program will do this for you as it copies the required device drivers to the user's Windows directory, when configuring each workstation). The last step should be to flag all files as shareable read only (SRO).

Some notes regarding the EXPAND program. If you notice the structure of the above batch file you will notice that the batch file will run the EXPAND program singly against each file on the diskette, rather than to try to expand them en masse one right after another. This is done for a reason. If you choose to simply issue a command like the following:

expand a:*.* f:\windows

You will not be successful. The first difficulty is that EXPAND does not give the user status as to what files it is uncompressing, nor how far along it is in uncompressing the file. While that is merely annoying, this combined with the second problem, could lead you to think that all files were processed, when in fact they weren't.

This is because the EXPAND.EXE program will process files on the diskette in directory order until all files are processed, or until it comes upon a file that is not compressed. If it does find an uncompressed file, it will stop and display a message saying that this file is not compressed. Unfortunately it will also stop processing the files on the diskette. This could leave some files, in directory order after the uncompressed file, that were not uncompressed and copied to the server directory. Currently with Windows 3.0 the only uncompressed files are SETUP.EXE, EXPAND.EXE, SETUP.INF and TOSHWIN.VCD.

Setup Windows User Files

Once you have setup the shared Windows directory on the server, you then need to configure the workstations. The first step is to run the SETUP program for each workstation. In order to install a network workstation set of Windows files, you need to run SETUP with the /N parameter. This instructs SETUP that this is a network workstation, and that the only files necessary to copy, will be the ones specific to this

workstation.

SETUP will then ask you where the personal Windows files will be located. Depending on your configuration you may elect to store these files in the user's personal directory on the file server, or on a local disk.

If you generate a SETUP error that says Cannot create WIN.COM, this usually indicates you ran SETUP /N with Windows files that have not been uncompressed. In order to fix this, uncompress all the Windows files with the EXPAND.EXE program and run SETUP /N again.

One of the choices that is given the user during the installation of Windows, is to build the Application Groups. If you choose to do this, and you click yes to the All Drives option, Windows will search all the attached drives starting at the root directory for any application that SETUP knows about. (The information on what applications Windows knows about is located in the SETUP.INF file) It will then build the appropriate groups, based upon what applications it found.

A problem here is that this utility will search all drive maps beginning at the root level, which, unless you are using the MAP ROOT option of the MAP command, would mean that the program will search and tag the same program files once for each drive map, that was mapped to the same volume. This increases the time the utility takes to run, as well as clutters up the final display of found programs.

A better strategy for configuring applications is to MAP ROOT a drive map to the highest point in the directory tree that you want the installation program to look through. Then just select that drive from the dialog box. This would obviously have to be done prior to entering the SETUP program.

If you did not MAP ROOT the appropriate drive map prior to entering SETUP, you can just cancel this choice and run this same utility from inside Windows. This is done from the Windows Setup Options Set Up Applications menu.

Figure 11 shows the files that SETUP will create and/or copy to the Windows user's personal directory. In addition to those listed, if the mouse used is a Microsoft or compatible one, SETUP will also copy MOUSE.SYS to the user's personal directory.

Startup Considerations

The following sections will deal with some startup considerations regarding running Windows on a NetWare workstation. These considerations are memory management, drive caching, Windows swap files, and network login issues.

Memory Management

One of the SETUP program choices involves the changing of your AUTOEXEC.BAT and CONFIG.SYS files. If the user files are being installed in a network directory, SETUP will create a AUTOEXEC.BAT and a CONFIG.SYS which will contain recommendations for how to configure the workstation's AUTOEXEC.BAT and CONFIG.SYS file, but it will copy those files into the selected network directory. If you are installing these files on a local drive, SETUP will in fact offer and then change these files for you. The AUTOEXEC.BAT changes are nothing more than including the new personal Windows directory in the PATH statement.

However, recommended changes to the CONFIG.SYS will usually have more affect on your system. SETUP will always want to install HIMEM.SYS and SMARTDRV.SYS into your CONFIG.SYS file. This will sometimes mean replacing current memory managers already installed in your CONFIG.SYS. (In fact there is a list of memory managers in the SETUP.INF file that Windows specifically looks to remove from CONFIG.SYS files due to the fact that they conflict with HIMEM.SYS.)

Before accepting these recommendations it is important to consider the following. The only memory manager that is required to run Windows in two of its three modes, is an XMS compatible one, version 2.0 or higher. Loading HIMEM.SYS will limit what further memory management you will be able to do. For

example, HIMEM.SYS does not allow you to make use of the High RAM (sometimes referred to as Upper or Adapter RAM) area between 640KB and 1024KB in the PC memory map, for loading TSRs or other drivers. Some third party products do provide alternative XMS solutions.

Currently Quarterdeck is shipping a combination Expanded Memory, XMS Extended Memory, and High RAM memory manager in its QEMM386 product. If you are currently using the 5.11 version of this driver, you do not need to load Microsoft's HIMEM.SYS. Qualitas, and All Computer, Inc., among others, also ship products which contain XMS 2.0 or higher, compatible memory management.

If you want to use an XMS only driver like Microsoft's HIMEM.SYS, and you want to utilize expanded memory for DOS applications, you must take care not to load memory managers that will conflict with one another. Microsoft's EMM386.SYS (on 80386 equipped PCs only) and its HIMEM.SYS can be loaded together.

Also, in regards to expanded memory and Windows, Windows and Windows applications only use expanded memory in real mode. In Standard and in 386 enhanced mode, Windows and Windows application use extended memory, via the XMS specification. Additionally, running in 386 Enhanced mode, Windows provides expanded memory support for non-Windows applications as part of Windows. No additional expanded memory manager is required.

Looking at your AUTOEXEC.BAT and CONFIG.SYS, as well as your user's needs you should consider the following in .

SMARTDRV.SYS

SMARTDRV.SYS is a Windows aware drive caching program. Depending upon how much memory is installed in your system, the SETUP program can be liberal in the amounts of memory that it recommends be allocated, for this program. Experimenting to see if there are significant performances changes using lesser amounts of memory, is advisable since the additional free memory will become useful, as you begin to use bigger Windows programs. If you already use another drive caching program, testing both caching programs for performance in and out of the Windows environment can be beneficial in determining the correct caching program to use. If you do not have a local hard drive from which you are performing Windows work, do not use a drive caching program. SMARTDRV will not cache network drives.

Swap Files

Windows will use temporary swap files for various operations such as spooled output files from the Windows Print Manager. The location of the temporary swap files will default to the location of the Windows startup files, unless explicitly told to swap elsewhere via the TEMP environment variable (or in 386 Enhanced mode, the PagingDrive= SYSTEM.INI setting). As in the following:

SET TEMP=C:\TEMP

This would use the directory TEMP on this workstation's drive C. Using network drives to perform this swapping activity carries certain penalties. Swap files that are swapped to network disks, not only creates more work load for the server, it creates more traffic over the network. Additionally, when you are placing swap files on an Advanced NetWare 2.15 server, there will be a marked increase in Windows load time. This is especially true when running Windows in 386 Enhanced mode.

The problem is that when a swap file is created, Windows will allocate a chunk of disk space for the swap file, prior to using it. When this is done on a NetWare server, NetWare will allocate the requested disk space, but prior to allowing the user access the file, NetWare will also zero fill the allocated file space. This is a NetWare operating system security measure and cannot be turned off.

The solution for problems resulting from using the network for swapfile location is to simply move the swapfiles to local devices. While a local hard drive is a good place for swap files, a local floppy disk drive should not be considered.

RAM disks are speedy places for swap files, however, it is recommended that you have a minimum size RAM disk of 2MB, for any configuration. The memory set aside for the RAM disk must always be considered against having this memory available to Windows applications. You also need to load a RAM disk device driver that will not conflict with any loaded memory managers. If you have further questions regarding RAM disks and Windows you can consult the Microsoft Windows User's Guide pages 530-535.

In 386 Enhanced mode you can specify a location for a permanent swapfile as opposed to temporary, on a local DOS device. This cannot be setup on a network drive at all. This is also outlined in the Microsoft Windows User's Guide on pages 520-530.

Logging in to the Network

It is recommended that users first login to their servers and then run Windows. While you are running Windows, you will be able to attach and detach servers, print queues, and drive maps all from within the Windows desktop.

Windows documentation suggests that if your network driver supports login and logouts from the Network Utility menu, then it is perfectly acceptable to do so. The Windows NetWare driver explicitly does not support this feature, at this time. In order to maintain integrity among the separate DOS and Windows sessions, you should not login or logout of a server from any DOS session.

Finally both the user's personal directory, and the shared server Windows directory, should be in the user's path (assigned as NetWare search drives). It is important that the user's personal directory precede the main Windows directory in the path. SETUP will suggest that, as a PATH statement in your AUTOEXEC.BAT. Since you will always be logging into your server before running Windows, these can be set up as search drives as opposed to set up in the PATH statement.

Windows/NetWare Configuration Files

Prior to logging into your network and running Windows there are some settings in certain configuration files which should be understood. These configuration settings reside in four files. The NET.CFG file, the WIN.INI file, the SYSTEM.INI file and the NETWARE.INI file. The following section will look in depth at the necessary Windows, NetWare and network settings contained in these files.

All configuration files are standard ASCII files and can be edited with any text editor, like Windows Notepad. It is important that if they are edited, the files be saved as standard ASCII text. Saving them in word processor's document format would render them unusable to Windows and/or NetWare. It is also recommended that when changing any configuration lines in the Windows .INI files that you first comment the original line out by using a semicolon at the front of the line, and then add the changed configuration line. For example changing the NetHeapSize in the SYSTEM.INI file would look like that pictured in .

SYSEDIT Utility

There is an undocumented Windows utility that can aid in viewing and editing certain startup files, called SYSEDIT. When this utility is run from the Windows desktop it opens four windows, and calls up a separate file in each window. The four files are WIN.INI, SYSTEM.INI, the workstation's AUTOEXEC.BAT, and CONFIG.SYS. These files can then be edited and saved with the SYSEDIT utility.

SYSEDIT is installed in the user's SYSTEM directory when a full version of Windows is installed, or it is located in the shared server directory in the case of network installations of Windows. SYSEDIT can be installed in any program group by simply following Windows procedure for doing so.

SHELL.CFG

NET.CFG is a configuration file that can either replace the SHELL.CFG or be used along with it. Some settings can only be used in the NET.CFG file. Later on in this Application Note, some of the TBMI settings, will be an example of these. However, for this discussion all of the following settings can go in either

configuration file. In order to determine which configuration files you should be using, and for a fuller description of the SHELL.CFG and NET.CFG configuration files, you can consult Appendix A and E from the NetWare 386 3.1 Installation Manual.

There are several NET.CFG settings that might be useful for running the Windows environment. The first concerns the way in which the NetWare shell handles the directory entries (.) and (..). To DOS, the (.) and the (..) represent the current and the previous directories in the directory structure, respectively. When using certain types of programs NetWare will not usually show these as directory entries. This can cause some problems for File Open dialog boxes in Windows applications. (It would not be possible to double click on the (..) entry to move one directory up in the path.) This is solved with the following NET.CFG setting:

SHOW DOTS=ON

This setting will insure that the shell will show the (.) and (..), as the current and previous directories to all applications. However, in order to use this option you must update the MAKEUSER and BINDFIX NetWare utilities to be the same or newer than those listed in Appendix A. Additionally, please consult the Novell Technical Bulletins in Appendix B regarding the BINDFIX utility.

NetWare's default of 40 open files per workstation, is also something that user's might need to increase. Microsoft recommends increasing that to 60 open files with the NET.CFG setting of:

FILE HANDLES=60

This should be consistent with the FILES statement in your CONFIG.SYS file (FILES in the CONFIG.SYS file and FILE HANDLES in the NET.CFG should always be consistent). This means that this user would need a FILES=60 statement in their CONFIG.SYS file.

Applications that use NetBIOS communications can sometimes require specific timing on NetBIOS broadcasts, (such is the case with 3270 terminal emulation software). Use the following values if there is a problem with NetBIOS sessions hanging:

NETBIOS BROADCAST COUNT=5

NETBIOS BROADCAST DELAY=10

WIN.INI

WIN.INI is the Windows Initialization file that primarily contains settings that allow a user to alter or customize their Windows environment or customize their Windows applications. This file is used to keep track of user and Windows application defaults as well as define the standard screen fonts which are display dependent.

There are basically nine sections to the WIN.INI file. They are shown in Figure 14.

The only specific change that Windows SETUP makes to the WIN.INI file, when configuring a NetWare workstation, is to specify a program to load upon Windows startup, in the [windows] section. The line that is added by SETUP is:

[windows]

load=nwpopup.exe

NWPOPUP.EXE is the program that translates NetWare message broadcasts into Windows format which is then displayed on the Windows desktop. One thing to remember about NWPOPUP.EXE is that even though you never see an indication that it is running or loaded, it is a running Windows application. This is important when running the Windows Swapfile application which must be run only when no other applications are running. (See pages 520-530 in the Microsoft Windows User's Guide regarding Swapfile.)

In order to disable the NWPOPUP.EXE program you can either remove it from the load= line in the WIN.INI

file, or you may Disable Broadcast Messages from the Network option of the Windows Control Panel.

SYSTEM.INI

The SYSTEM.INI is the second important Windows initialization file. This file is used to define and customize Windows for your hardware, or system configuration.

It is extremely important that you remember that changing settings incorrectly in the SYSTEM.INI file can result in a Windows system that will not run. Before making changes, consult the appropriate Microsoft manuals and/or SYSINI.TXT files. Also, make changes by commenting out the original line with a semicolon at the beginning of the line, and then repeating the entire line after the original, with your new settings. This way you can change values and still have the original setting listed in the file.

There are six basic sections in the SYSTEM.INI file. They are:

(In any of the Windows configuration files, if you want to enable a Boolean setting, you can enter: true, yes, on, or 1. If you want the Boolean setting to be disabled, you can enter: false, no, off, or 0.)

Following, is a more in-depth discussion of the NetWare section settings.

[NetWare]

RestoreDrives=<Boolean>

Normally when you exit Windows, all of your drive mappings are restored to the way they were before you started Windows, and all changes you made inside Windows are lost. RestoreDrives is the setting that determines this. The default condition is true, which means all drive maps are restored to their previous state before entering Windows. This means any drive maps made while within Windows will be lost upon exiting. If you set the RestoreDrives value to false, the mappings you made inside Windows will remain when you exit Windows.

[NetWare]

NWShareHandles=<Boolean>

As stated previously, in Windows 386 Enhanced mode each DOS session that gets started whether as a DOS command prompt or as a non-Windows applications, is created using the virtual 8086 mode of the 80386. This creates a separate virtual machine for each session.

Also each virtual machine you start in Windows gains its own set of drive mappings. It copies this information from the global memory of the System VM. However, the default method for doing this is to copy this drive mapping information into the local memory of the specific virtual machine. This means that changes to one virtual machines drive maps are not reflected throughout all virtual machines. Mapping changes are local to each VM.

NWShareHandles allows you the option of maintaining the drive map information in the global memory of the System VM. This would mean that changes in any VM's drive maps would be instantly reflected in all other VMs.

Other SYSTEM.INI settings that can affect network operations are listed in and . Appendix C provides detail on each of these setting as described from the Windows supplied text files SYSINI.TXT, SYSINI2.TXT, and SYSINI3.TXT. This detailed description explains each setting in regards to what it does and how it can affect network operations. Additionally, default, changing and suggested values, are given for each setting, where applicable.

NETWARE.INI

This third Windows initialization file, is exclusively created for NetWare users. This file is not on any NetWare or Windows distribution disk, but instead is created by the NetWare Windows driver,

NETWARE.DRV, whenever the driver cannot find a current copy of the file. The NETWARE.INI file contains three kinds of information.

The first kind of information, is information regarding the utilities available to the Windows user through the network driver. In this case shows the standard NETWARE.INI as created by NETWARE.DRV.

Using the syntax UTILITY=NAME, this shows that the NetWare Windows driver provides four utilities. The ability to attach to a server, detach from a server, to enable receiving messages and disable receiving them. Note that the syntax is composed of a description of the utility, which shows up as the choice in the Control Panel Network list box, an equal sign, and an executable command name. The < character preceding the command name, signifies an internal command, which is internal to the NetWare driver.

Fig. 19 shows the Windows NetWare network utilities main menu, which is accessible from the Control Panel Network selection.

The second type of information available in the NETWARE.INI file is environment settings. The current options available are listed in Figure 20. These settings must go under the heading of [MSW30-PrtQ] as shown.

These environment settings control how the Windows Print Manager keeps track of, and displays information about print jobs. MaxJobs is a setting which controls how many print jobs are visible in the Windows Print Manager queue. MaxJobs has a range 1-250 with a default setting of 50. MaxBufSize is the companion setting to MaxJobs. This setting configures the size of the buffer in bytes, that Windows will use to store information about print jobs in the Print Manager queue. The MaxBufSize range is 3500-30000 with 3500 being the default setting. UpdateSeconds is simply the time interval that the Print Manager automatically checks and updates the Print Manager queue. UpdateSeconds range is 1-65 with the default at 30. This function can still be done at any time manually, via the Print Manager menus.

The third kind of information that can be contained in NETWARE.INI is user again user configurable. This can be other programs which need to be accessed from the Control Panel Network menu. The following are some examples of lines that can be added:

Figure 22 shows the Windows NetWare network utilities drop down selection menu, which is accessible from the Control Panel Network selection after selecting the down arrow button. This drop down menu shows the four internal NetWare commands from the NETWARE.INI file. This menu also shows the external commands listed in the sample, NETWARE.INI file.

Runtime Considerations

The following are some runtime considerations for running Windows on a NetWare workstation. The first is something mentioned previously in this Application Note, and it regards the use of the newly added, MAP ROOT ability. The second run time consideration is in regards to integrating Windows printing output with NetWare print spooling.

Map Root

Most users are familiar with the fact that drive mappings in NetWare, prior to the addition of the MAP ROOT command, would map a drive letter to a drive/directory combination beginning at the root level of the target volume. This means that when you created a drive map like the following:

MAP F:=FS1/SYS:USERS/GUEST

You would assign drive letter F: to point to the directory GUEST a subdirectory of USERS, which exists at the root level of the SYS volume on server FS1. If you then changed to F: and executed the DOS command to change a directory to the next level up, CD .., you would change your drive F: pointer to point to the next directory level up, in this case the directory USERS.

As a result, a problem that is created, is with applications that ignore default directories for a drive letter. In

particular, the Windows File Manager resets the default directory for a drive letter to the root level. Using the above example the File Manager would reset drive F: to point to the root level of volume SYS on server FS1. Another example where this happens is when Windows scans the connected drive letters for the purpose of building application groups. This is done during the SETUP program, or once in Windows, by running the Windows Setup Options Set Up Applications utility.

Drive A: maps to a local disk.

Drive B: maps to a local disk.

Drive C: maps to a local disk.

Drive D: maps to a local disk.

Drive E: maps to a local disk.

Drive F: = FS1\SYS: \USERS\BILL

Drive G: = FS1\SYS: \USERS\BILL\WP

SEARCH1: = Z:. FS1\SYS: \PROGRAMS

SEARCH2: = Y:. FS1\SYS: \PROGRAMS\WIN

SEARCH3: = X:. FS1\SYS: \PROGRAMS\WP51

SEARCH4: = W:. FS1\SYS: \PUBLIC

Given the following drive maps as an example, running the Windows Setup Options Set Up Applications utility and selecting All Drives to scan, would cause the utility to scan the SYS: volume and all subdirectories (which you had sufficient privileges to scan) six separate times. This would also build a list of applications that Windows knows about, six times larger than actual, since each location of the application file would be defined by the drive letter identifier.

In order to force applications not to reset default directories on drive letters, the updated MAP.EXE program and NetWare shells (consult Appendix A for listings of these files) were modified to allow the use of fake roots. Reissuing the previous example drive mapping with the MAP ROOT command looks like the following:

MAP ROOT F:=FS1/SYS:USERS/GUEST

This would assign the drive pointer to the same volume/subdirectory location as in the previous MAP command, however this time when you changed to the F: drive and issued the DOS command, CD.., the drive map and directory location would not change. To all applications and DOS, this drive letter would look like a root directory. The difference between both types of drive maps can be noted when you display your drive maps with the MAP command. The following is an example of drive maps and search drive maps issued by both the MAP and MAP ROOT commands.

MAP Drives

Drive F: = MS1\SYS:USERS\BILL\WP

SEARCH1: = Z:. [MS1\SYS: \USERS\BILL\WIN3]

MAP ROOT Drives

Drive F: = MS1\SYS:USERS\BILL\WP \

SEARCH1: = Z:. [MS1\SYS:USERS\BILL\WIN3 \]

Since the Windows File Manager program resets drive identifiers to their root level, at the minimum it is recommended that you use MAP ROOT drive mappings for all search drives and for regular drive maps that are used consistently. One issue to watch out for when you begin using MAP ROOT is applications that have relied on being able to explicitly state a complete path based on a previous drive letter mapping.

NetWare Printing

Windows is best described as an operating environment. This is because Windows controls some aspects of the PC's operating environment that were previously controlled by the operating system. The handling of output devices is one of those areas.

The NetWare shell is considered an operating system extension because it extends normal DOS functions across a network. As in the case of spooled printed output. The situation this creates is one in where the way in which Windows hands off the printed output must be integrated with the way NetWare spools the output to the shared printers. Whether print jobs are defined in PRINTCON for all users, or whether all users use separate CAPTURE or NPRINT statements to send output off to network printers, there are four options that need to be set in order things to work right. These are shown in Figure 23.

The reasons for these settings are as follows. First, Windows always sends a form feed after a print job in much the same way as some standard DOS applications do, so this feature can be turned off for NetWare.

NetWare will perform tab expansion, (converting a tab character to 8 blank spaces), for output when told to do so and it is in fact the default method for handling printed output. However, almost all Windows output is in bitmap form. The problem is that the byte values resembling tab characters can occur randomly in any bitmap output. With tab expansion on, this would cause the print job to be expanded with blank spaces wherever NetWare encountered a byte value for a tab. At the printer this would create seemingly random blank spaces in the output. For this reason it is necessary to tell NetWare to perform no tab expansion. This is done by specifying the file contents as Byte Stream in print job configurations and by using the NT switch with CAPTURE and NPRINT.

Turning off the automatic endcap and the timeout is necessary due to the fact that Windows can exceed the NetWare timeout periods in the production of some print jobs. This problem would be shown by receiving partial print jobs at the printer.

Other NetWare printing options are not necessary for Windows printing.

An example of the CAPTURE and NPRINT commands that contains all the necessary switches would be the following:

CAPTURE NFF NT NA TI=0

NPRINT filespec NFF NT NA TI=0

NetWare Non-Windows Application Support

As mentioned earlier in this Application Note, Windows handles non-Windows Applications by either task switching these sessions in and out of conventional RAM space in Real and Standard mode, or by running virtual 8086 sessions in 386 Enhanced mode.

A problem was recognized in regards to applications that accessed the network as they were run from within Windows. Novell has since added improved support for non-Windows applications running under both Standard and 386 Enhanced mode. The problem that was corrected was first recognized under the task switched environment of Standard mode.

As mentioned in the opening sections of this Application Note, when Windows in Standard mode performs

its task switching it maintains global memory which is not switched out of the lower 640KB of RAM, and which contains system information such as the COMMAND.COM processor, Terminate and Stay Resident (TSR) programs, and network or other drivers. This global memory is used as system resources for each successive switched session.

Operationally this provides a single global memory segment that is never swapped out of the lower 640KB of RAM, and separate switchable local memory segments (one for each session running) that are swapped in and out of the lower 640KB of RAM.

Running most applications on a NetWare LAN in this environment, would not usually cause a problem. This is because the NetWare shell that is loaded, intercepts all calls a DOS application makes and then makes the determination whether to pass it along to the network or to the local PC. The very same network shell is one of the pieces of system information located in the Windows global memory. Because this memory never gets switched out of conventional RAM, the shell can correctly buffer and coordinate network communications among the various switched sessions.

However there are a number of non-Windows network applications that communicate with the network by directly issuing IPX/SPX calls, forgoing the NetWare shell. In the aforementioned task switched environment this would create a situation where an application in local memory would make calls directly to the IPX/SPX driver located in global memory with the danger that before the communications can complete, the current local memory contents would be switched for another application. When this happens the IPX/SPX driver will lose contact with the calling application and not be able to use its data for network communications.

It is for this type of non-Windows applications that the Task switched Buffer Manager for IPX/SPX (TBMI) was developed.

Task switched Buffer Manager for IPX/SPX (TBMI)

The Task switched Buffer Manager for IPX/SPX (TBMI) is a combination of two programs that serve to buffer and synchronize direct IPX/SPX network communications between the network and applications that operate in a task switching environment. Such as Windows in Standard mode.

In order to accomplish this, TBMI is loaded prior to running Windows. By doing this, TBMI installs itself in what becomes the global memory segment for Windows. Situated there TBMI intercepts calls to IPX/SPX from applications running in the local memory segments. TBMI then buffers each separate IPX/SPX communications for each calling application. The second portion of TBMI is loaded with each non-Windows application (that communicates directly with IPX/SPX) and is called TASKID. This memory resident program passes task identification information to TBMI which allows TBMI to map each network communication with the correct calling application. Both TBMI and TASKID components are necessary for proper synchronization of network communications to take place.

TBMI is not needed for both Windows applications and for non-Windows applications that utilize the NetWare shell for network communications. Windows applications utilize a different mechanism to access IPX/SPX network communications and non-Windows applications that utilize the NetWare shell for network communications are serviced similar TBMI functions as part of the shell's duty. Lastly, there is no need for TBMI in any type of non-Windows application session if you will not be switching between sessions.

Finally if your are unsure as to whether or not you need to use TBMI or not, the TBMI.DOC file included with the TBMI files states the following.

If you are not sure that your applications make direct call to IPX/SPX, go ahead and run TBMI. It will only affect operations by using up a small amount of memory, After running the application for a period of time, enter the command TBMI /D and look at the count values associated with the fields `Far Call Usage' and `Old Int Usage'. These values specifies how many times TBMI has actually received calls. If these values are zero, then your application is not using TBMI and you may safely run the application without it.

The following is the statistical output from issuing the TBMI /D command.

Task Switched Buffer Manager for IPX/SPX - version 1.0

(C) Copyright 1990, Novell Inc. All rights reserved

TBMI is currently resident

Interrupt 2Fh hooked

Interrupt 64h hooked

Interrupt 7Ah hooked

TBMI Buffers in use : 0000

TBMI Max buffers used : 0000

TBMI Unavail buffer count : 0000

TBMI Old int usage count : 0000 u (This value)

TBMI Far call usage count : 0000 u (And this value)

TBMI Next ID number avail : 0001

TBMI Most recent ID number : 0000

TBMI Outstanding ID count : 0000

TBMI Configured ECBs : 0014

TBMI Configured Data ECBs : 003C

TBMI Configured sockets : 0014

TBMI Current sockets : 0000

TBMI.COM

The following are the instructions and configuration parameters for running TBMI, from the TBMI.DOC file included with the program files.

This program provides the data buffers needed to virtualize the IPX and SPX requests made from applications running in a DOS session. Because these buffers need to be allocated before Windows starts, TBMI must be run before starting Windows. Figure 24 shows the valid TBMI command line parameters.

This program reads configuration information from a configuration file in the current directory. One parameter is entered on each line in the configuration file. This file's name is NET.CFG by default; a different name can be specified using the /C parameter on the command line. Figure 25 shows the valid configuration file parameters.

TASKID.COM

The TASKID.COM program must be run in each DOS session before an application is started in that session that requires direct IPX or SPX support. This provides the two-way communication needed for virtualizing asynchronous events in the DOS session. This program provides a unique ID to TBMI which is used in virtualizing IPX and SPX calls.

Fig. 26 shows the valid command line parameters.

It is recommended that you unload TASKID from the DOS session's memory before closing the session. Do not unload TASKID before task switching.

TBMI Usage

Do the following to start TBMI.

1. Start TBMI by entering the command TBMI on the command line, followed by optional command line parameters listed above.

2. Start Windows normally.

3. Start a DOS session by clicking on the DOS Prompt icon. At the new DOS prompt, enter the command TASKID followed by optional command line parameters listed above.

4. Repeat step #3 for each DOS prompt you open before running an application from that prompt.

Before closing a DOS session with the EXIT command, remember to first type "TASKID /U" to unload TBMI from that session. Failure to unload TASKID from a session's memory before closing the session may result in loss of data buffers and machine hanging. It is not necessary to unload TBMI after exiting Windows, but you may wish to do so to free up memory.

TBMI could be included in a batch file starting Windows to insure that it is always started before Windows and unloaded afterwards. For example, the batch file WIN.BAT could include the following:

TBMI

WIN

TBMI /U

A batch file can also be created to automatically load and unload TASKID when starting and exiting a DOS session. The batch file must be specified instead of COMMAND.COM in either the Properties box or the PIF file of the DOS Prompt icon. The batch file should include the following lines:

TASKID

COMMAND

TASKID /U

Windows 386 Enhanced Mode and TBMI

It is recommended that once TBMI is installed on your system that you remove the automatic loading of VIPX.386 as TBMI also replaces the functionality provided by the 386 Enhanced mode driver. This is done in the SYSTEM.INI file under the [386Enh] section, as shown in Figure 27.

NetWare Utility Program Information Files (PIF)

The last issues in regards to running non-Windows applications from within Windows concerns NetWare utility programs. In keeping in line with Microsoft recommendations, Novell recommends setting up PIF files for running NetWare utilities. This provides the primary benefit of locking down system resources for use by the NetWare utilities.

Figure 28 shows the suggested PIF settings for NetWare utilities as listed in the example NETWARE.PIF file for Windows in Standard mode.

Figure 29 shows the suggested basic PIF settings for NetWare utilities as listed in the example NETWARE.PIF file for Windows in 386 Enhanced mode.

Finally, Figure 30 shows the suggested advance PIF settings for NetWare utilities as listed in the example NETWARE.PIF file for Windows in 386 Enhanced mode.

For further information regarding PIF files consult the Microsoft Windows User's Guide pages 440-490.

Advanced Windows LAN Administration

In setting up multiple Windows workstations, an issue that has been raised is in regards to setting up the Windows configuration files so that users might be able to login to the network and run Windows, all from various differently configured workstations. Windows does not lend itself easily to this, but with a little configuration it is possible to accomplish. In order to do this the Windows users' files must be located on network drives (which raises the issue of swap file locations but that's discussed elsewhere in this document). Also, changes need to be made to both how the SYSTEM.INI and the WIN.INI files are set up.

Since the hardware configuration information is located in the SYSTEM.INI file it is necessary to maintain a common set of SYSTEM.INI files for either each workstation, or for each class of workstation. Then based upon a system variable at run time a batch file can copy the necessary SYSTEM.INI file to the user's Windows directory. The system variable used as an example here is SYS_TYPE.

If the network is made up of distinct classes of workstations then environment variables could be used to identify each class. For example for all Compaq 386es with VGA displays could be assigned the environment variable:

SET SYS_TYPE=CPQ386V

For all IBM 286 PS/2s with VGA the environment variable could be:

SET SYS_TYPE=IBMPS22V

SYSTEM.INI files for each class of machine could be created by running SETUP on one of the machines and then copying the SYSTEM.INI file to the common SYSTEM.INI directory, and renaming based upon the system type. Such as CPQ386V.INI and IBMPS22V.INI.

A Windows batch file could then consist of the following commands:

```
ECHO OFF

COPY  F:\WINDOWS\SYSINI\%SYS_TYPE%.INI   F:\USERS\BILL\SYSTEM.INI

WIN
```

However, in order for this to work properly the WIN.INI file, which maintains most of the application and environment specific information needs to have one section modified. That is that the WIN.INI files for all users should have all screen fonts installed for all possible monitor types.

This is because the screen fonts section of the WIN.INI file is the only hardware dependent part of this file. This allows each user to maintain a unique WIN.INI file which would contain their own application specific and environment information regarding the user and application defaults used by Windows and Windows applications.

shows what the [fonts] section of such a WIN.INI file might look like. This WIN.INI file would be configured for CGA, Hercules Monochrome (which uses the EGA fonts), EGA, and VGA displays.

Additional advanced Windows LAN management topics are covered in the following publications:

Graphic Facts About Networking with Windows. Automated Design Systems, Inc. 1988-1990.

Saber LAN Setup Guide for Windows Version 1.0. Saber Software Corporation 1990.

Troubleshooting

This section covers common problem areas and common technical support questions and answers. The first part provides a history of the NetWare DOS shell and the problems that have been corrected with the shell revisions. Following that will be a break down of other Windows / NetWare problems and solutions by category of problem. Some of these categories are initial setup, application, Windows, and NetWare problems.

If after consulting this section you find you have problem that is not covered here you may pursue the problem with your local NetWare dealer. If you still cannot resolve the problem you can contact Novell Technical Support at 1-800-526-7937 (1-800-LAN-SWER). This service is a charge service.

NetWare DOS Shell History

The following is a historical description of the problems and solutions to various NetWare DOS shell issues for the NetWare DOS Shell v3.01. This history file is included in the latest version of the NetWare DOS shell. This document is also updated with each subsequent release of the NetWare DOS shell.

Loading SiteLock by Briteworks would fail, causing the DOS workstation to hang. This problem was corrected with the 3.01 rev B shell.

Using the Preferred Server option caused the network response time to be functionally slower than if the user did not use this option. The 3.01 rev C shell corrected this problem.

When using DOS 4.0 with EMSNETx and XMSNETx shells the DOS directories would not display correctly under Windows. This was corrected with the 3.01 rev C shell.

The enhanced memory shells were not sending header information when using job configurations that included escape codes. For example, a job that should print landscape would print using the default mode (portrait).

When printing to a captured LPT device an error message Device not ready would appear. A retry would allow the job to continue. The 3.01 rev C shell corrected this problem.

Fake roots were being deleted on paths with volume names before the path was determined valid such as CD PRN: would delete the fake root. This was fixed with the 3.01 rev C shell.

On 286-based servers the Dynamic Memory Pool (DMP) 1 was not being released properly with the XMSNETx and EMSNETx shells causing the server to hang eventually. With the 3.01 rev C shell the memory is released when exiting the Windows DOS Prompt.

The NetWare DOS Shells Rev. C was made available to NetWare Developers only. The NetWare DOS shells v3.01 rev D was released to all users and contains all the 3.01 rev C changes.

When running the 3.01 rev D shell on a NetWare V2.15 or less operating system, external program execution (using the #) from the login script does not work unless the user has open privileges at the volume root. This has been corrected in the shells dated 9/18/90 or later.

Nver will return Rev. C instead of Rev. D. This has been corrected in the shells dated 9/18/90 or later.

When using the DOS 4.0 TrueName (undocumented DOS command) command invalid data was returned to the shell. This invalid data causes Emerald's System's backup to not function properly. The 3.01 rev E shell corrects this problem.

Microsoft Link was reporting a scratched file error when linking a large number of files. This was corrected in 3.01 rev E of the NetWare DOS shell.

Added support for VERSION.EXE utility. This support was not present in earlier releases of the shell.

Corrected a problem with the rename function where the wrong error code would be returned to applications such as Platinum Accounting by Advanced Business Microsystems. This error was also exhibited with the NETGEN message: Cannot find DRVRDATA.DAT.

Corrected a problem where the shell was not correctly maintaining the default server after logout when an X.25 bridge is used.

On ELS NetWare servers you would get one less connection than the maximum when using remote boot. The 3.01 rev E shell corrected this problem allowing the user to get all connections to the server.

Enabled file caching in EMSNETx and XMSNETx shells. File caching was not enabled in earlier releases of the enhanced memory shells. Also fixed a problem where these shells were passing an incorrect file server address to IPX. The error most commonly seen was No response from server <servername>

Added the /? option to the command line which displays version and usage information.

Added a feature in the 3.01 rev E shell that tells the user that a TSR is loaded when trying to unload the shell.

DOS Issues

Problem: DOS 4.0 and Windows Generic Problems

Explanation: Some problems seem to exist when running Windows and using DOS 4.0 ANSI.SYS and SHARE.EXE.

Solution: Try removing ANSI.SYS from DOS startup files.

Microsoft recommends in the NETWORKS.TXT file to try removing SHARE.EXE from DOS startup files. (Some versions of DOS 4.0 and DOS 4.01 automatically load SHARE.EXE when using single drive sizes of greater than 32mb. To disable SHARE.EXE from loading you can rename the program file.)

NetWare Issues

Problem: SHGEN goes from A: to B: endlessly

Explanation: This is due to the fact that the SHGEN.EXE program will look for one of two things when it runs. It will either look to see if it being run from a diskette labeled SHGEN-1 and if it is, it will proceed to look for all the necessary files in the root level of this diskette. (These would be the files from the DSWIN2.ZIP and the IPX.OBJ from DSWIN1.ZIP archives) Or SHGEN.EXE will look for a subdirectory at the default directory level called SHGEN-1 for the necessary files and optionally look for directories labeled LAN_DRV_.??? for LAN driver files.

Solution: Put all the files from DSWIN2.ZIP and the IPX.OBJ from DSWIN1.ZIP, on a floppy labeled SHGEN-1 and then run SHGEN.EXE. Or create a subdirectory called SHGEN-1 and put all the files from DSWIN2.ZIP except SHGEN.EXE in that directory along with the IPX.OBJ from DSWIN1.ZIP.

Problem: Can't figure out which NetWare diskettes the DSWIN files go on

Solution: The files should be put on the working copies of your NetWare diskettes as follows:

DSWIN1.ZIP	SHGEN-1
DSWIN2.ZIP	SHGEN-2
DSWIN3.ZIP	UTIL-1

Problem: Novell Menu System returns Cannot find beginning of menu file type error message when exiting Windows

Explanation: Windows expects to be the only menu system and not be run from another menuing system.

Solution: The Novell Menu system can be used if the menu utility and overlays are copied to a local drive, and executed from there. Alternatively several third party menu utility vendors will operate correctly in this situation.

Problem: Receiving EMS memory errors

Explanation: The EMS NetWare shell was very strictly written to the LIM/EMS 4.0 specification. When the Expanded Memory Manager returns an error, the EMS NetWare shell simply returns that error to the screen with the notation that the error has been returned. Any errors of this kind are a result of a non-LIM/EMS 4.0 compatible driver.

Solution: Upgrade EMS driver to latest LIM/EMS 4.0 compatible version.

Problem: Not enough XMS memory to run Windows in Standard or Enhanced mode

Explanation: After using the NetWare XMS shell, there was not enough XMS memory left over for Windows to run in either standard or 386 Enhanced mode.

Solution: Add more memory or use either the conventional or expanded memory NetWare shell.

Initial SETUP Issues

Problem: SETUP Program hangs when loading

Explanation: Some types of systems can cause the Windows SETUP program to hang when SETUP attempts to read the system information.

Solution: Run SETUP with the /I switch.

Problem: SETUP Program hangs when loading (continued)

Explanation: Windows uses the I/O address 2E0h for detecting an 8514 video adapter. Some NICs will also use this I/O address and cause a conflict with Windows

Solution: Change the NIC's I/O address.

Problem: SETUP Error: Cannot create WIN.COM

Explanation: You will usually get this error when you run SETUP /N and the Windows files have not been uncompressed.

Solution: Uncompress all the Windows files with the EXPAND.EXE program and run SETUP /N again.

Problem: Error: Update network shell type error

Explanation: This error is caused by not using the 3.01 or higher version NetWare shells.

Solution: Use the 3.01 or higher version NetWare shells.

Windows Issues

Problem: Windows hangs when loading

Explanation: Windows may hang for a number of reasons.

The first thing that can be done to isolate this type of problem is to attempt to run Windows in any of its three modes.

WIN /r runs Windows in Real mode.

WIN /s runs Windows in Standard mode.

WIN /3 runs Windows in 386 Enhanced mode.

If Windows refuses to run in 386 Enhanced mode but will run in Standard and Real modes, look for problems in the [386Enh] section of the SYSTEM.INI file. Problems such as conflicting page frame addresses, and not enough memory, can be the problem.

If Windows refuses to run in both Standard and 386 Enhanced mode then problems could exist with your XMS memory manager. Look for conflicts there.

If Windows refuses to run in real mode this usually indicates setup or configuration problems. Check the setup and configuration, especially available memory.

Problem: Windows hangs when loading

Explanation: Windows may hang for a number of reasons.

Windows' default EMS page frame is at D000-DFFF. If the base memory address of an installed adapter card overlaps this memory area, you can have problems.

Solution: Either change the base address of the affected card, or use the EMMExclude statement in the SYSTEM.INI file to exclude the memory segment in question. The following are some possible conflict areas:

Some 16-bit VGA Cards C400-C7FF

IBM Token Ring Card ROM 8KB CC00-CDFF

IBM Token Ring Card Shared RAM 16KB D800-DBFF

Problem: Windows hangs when loading (continued)

Explanation: NICs that use interrupt 2 (IRQ 2) or interrupt 9 (IRQ9) or greater, can cause problems when running Windows in 386 Enhanced mode.

Solution: 1) Change the NIC's interrupt.

Solution: 2) Use the VPICDA.386 driver (available from Novell) in place of the VPIC driver in the [386Enh] section of the SYSTEM.INI file.

Problem: Windows hangs when loading (continued)

Explanation: Some PCs treat the COM ports as a family. If you have a serial mouse on COM1, using IRQ4 and your NIC is set to IRQ 3, it may hang being unable to access the network. This is especially true of Ethernet cards.

Solution: Change the NIC's interrupt level.

Problem: Windows hangs when loading (continued)

Explanation: If you are using the XMSNETx or EMSNETx shells, there is a memory allocation bug with Windows 3.0. This is typified by Windows hanging when loading.

Solution: You can get around this problem by using the conventional memory NetWare shell, NETx. Alternatively you can upgrade to Windows 3.0a which corrects this problem.

Problem: Can't find Group (.grp) files

Explanation: This error usually occurs as a result of mis-mapped drives. The specified location of the group files is in the PROGMAN.INI file. If the user has changed drive mappings from what was referenced in PROGRAM.INI, it will result in this error message.

Solution: Correct either the drive map assignments to reflect the values in PROGMAN.INI or change PROGMAN.INI values to correspond to the drive map assignments.

Problem: Group files are corrupt

Explanation: This seems to be mostly a Windows problem. If the group files become corrupted and they cannot be used, they must be reconstructed.

Solution: Delete the old group files and restore them from a good backup or rebuild them by hand.

Hardware Issues

Problem: Problems using MAP ROOT with some ethernet cards.

Explanation: There are some problems using older Western Digital Ethercard Plus cards and drivers, Everex ethernet cards and drivers, and the new MAP ROOT command.

Solution: Contact the respective manufacturers for updated drivers.

Problem: Problems running Windows with SMC Arcnet NICs and SMC Turbo Drivers

Explanation: There are some problems running Windows and NetWare using older SMC Turbo drivers with SMC Arcnet NICs.

Solution: Contact SMC for updated Turbo II drivers.

Application Issues Btrieve Requester

The following information is provided from Novell Development Products Division and is included in files used to run Btrieve under Windows 3.0.

The use of BREQUEST.EXE in Windows 3.0 is made possible by a DLL which acts as an interface between the application and the requester. The DLL makes use of the DOS Protected Mode Interface (DPMI) provided by Windows 3.0 to access the requester which runs in REAL mode.

The DOS Btrieve requester must be version 5.15 or greater. Previous versions of the requester will not work properly with the DLL.

The user must load the DOS Btrieve requester prior to starting Windows, and the DLL must be locatable by Windows. Access to local files is possible if you have the client based version of Btrieve for Windows by renaming the Btrieve DLL to WBTRLOCL.DLL and specifying local=yes as noted below.

The number of file servers (/S:) and redirected drives (/R:) options used to initialize the Btrieve requester must be large enough to accommodate additions after loading.

If both DLL's are used, a Btrieve VERSION operation will return both versions if the data buffer length is at least 10. The Btrieve requester version information will be immediately followed by that of the client based Btrieve.

Additional status code: 1013. This status is returned when the DLL's task list is full. The number of tasks can be increased up to 255 as specified below. The most common cause of this problem is that the application does not successfully complete a call to WBTRVSTOP() before terminating while other Btrieve tasks are active. This causes the task entry to remain and eventually will fill up the table.

The DLL needs to know a couple of things for initialization. Set in WIN.INI (defaults):

[brequestDPMI]

datalength=4096 ; same as the /D: option used for DOS requester

chkparms=no ; validate pointer parameters

local=no ; access local files through Btrieve DLL

tasks=10 ; maximum of active tasks to use the DLL

The datalength option should be the same as the /D: used to initialize the DOS requester when it is loaded. This should be at least as large as the largest record to be accessed.

NetWare 3270 LAN Workstation

What follows is the 3270READ.ME file that covers running the NetWare 3270 LAN Workstation program with Microsoft Windows. It is also included in the TBMI.ZIP file available on NetWire. This document as the 3270READ.ME file, is entitled Guidelines for Using the NetWare 3270 LAN Workstation with Windows 3.0.

Guidelines

The following guidelines enable the NetWare 3270 LAN Workstation for DOS to run in a Windows 3.0 environment. It should be understood that the NetWare 3270 LAN Workstation for DOS is a DOS application, and, as such, complies with DOS limitations with respect to the Windows environment. In addition, running an SPX application which makes direct calls to IPX or SPX under Windows requires the use of the TBMI and TASKID programs.

Novell is supplying these guidelines as an interim solution prior to the release of the NetWare 3270 LAN Workstation for Windows, which will be a true Windows-based application. Please note that if you encounter problems which appear to be workstation-related, we would like for you to reproduce the problem in a non-Windows environment prior to calling Technical Support.

Installation Tips

Before loading Windows, you must remove VIPX.386 from the NETWORK= statement in the SYSTEM.INI Windows file.

Use the following sequence to load the software:

IPX (use IPX.OBJ v3.02)

NET3 or NET4 (optional)

Login to file server (optional)

TBMI

WIN (from Microsoft Windows diskette)

Double click on the DOS icon and load TASKID, followed by WSLAN

We recommend that you enable the NetWare 3270 LAN Workstation to run in background and full-screen modes. Attempting to run the workstation software in a window may cause unpredictable results.

Because Windows 3.0 uses some of the same key sequences as the NetWare 3270 LAN Workstation, you will need to redefine them in order to avoid conflicts. Specifically, the <Alt><SysRq> and <Alt><Esc> (if configured for host printing) key combinations are used by both products. These key sequences may be redefined in either the DEFAULT.PIF file for the DOS session in Windows or by using the KEYDEF utility for the NetWare 3270 LAN Workstation.

If you would like to create an icon to execute a batch file to load WSLAN, the following sequence of commands is recommended:

TASKID

WSLAN

PAUSE

JUMP

WSEXIT

TASKID /U

EXIT

Note that the above is useful for those who need a single host session only and do not wish to run Send and Receive or API applications. This is because when you jump back to the DOS session, WSLAN must be unloaded in order to prevent the host session from hanging. However, we hope that this .BAT file illustrates the options available to customize your Windows environment.

Limitations

1) You must run in 386 Enhanced mode. We recommend that your machine have at least 4MB of memory.

2) You must use LAN printer redirection for printing from the host session.

3) SNA must use either Definite Response (set in the bind image) or Pacing (set to a non-zero value in VTAM) if you do not set the window session for background processing.

4) Closing the host window task without unloading the workstation software will cause the host session to hang. This will prevent you from getting back into the host session. Therefore, it is recommended that you run WSEXIT, followed by TASKID /U, prior to closing the host window.

5) A limitation of Windows is that it may not reload special fonts when you switch between workstation sessions. This causes the host status line to appear corrupted if your host session is configured for Extended Data Stream. Simply jumping to DOS, then back to your host session, causes WSLAN to automatically reload the status line font.

6) Sometimes Windows needs to be reloaded several times when you have a TSR in place. If, when you load Windows, you are taken back to the DOS prompt, simply reload Windows.

7) The mouse is not supported in a host session.

8) Vector Graphics is not supported in a host session.

NetWare Asynchronous Communication Server

What follows is the Novell Communications Products Division technical bulletin regarding operating the NetWare Asynchronous Software Interface (NASI) software while within Windows 3.0. This file is included in the NetWire ZIP archive PTF234.ZIP.

PTF 234: TBMI Windows DOS Compatibility Box for NACS

APPLIES TO: NetWare NACS

SUPERSEDES: NA. However, the TBMI and TASKID are the same as in TBMI.ZIP

DATE: January 4, 1991

If you encounter problems which appear to be NACS/NASI related you must reproduce the problem in a non Windows environment prior to calling Technical Support.

TBMI (Task switched Buffer Manager for IPX/SPX) and IPX.OBJ v3.02 correct problems running an application that makes direct calls to IPX or SPX (called a peer-to-peer application) under NetWare within a DOS box in a multitasking environment such as Windows. With TBMI and TASKID loaded, you will be able to switch away from a running peer-to-peer application (such as NACS) without problems.

Installation

1. See Limitations below
2. Generate IPX.COM from the v3.02 IPX.OBJ.
3. Remove VIPX.386 from the NETWORK=statement in the SYSTEM.INI Windows file.
4. (Optional) Create one or more configuration files. TBMI reads configuration information from a configuration file in the current directory. Enter one parameter on each line in the configuration file. The file's name is NET.CFG by default; a different name can be specified using the /C parameter on the command line.

Running

1. Load the new IPX
2. Load NET3 or NET4
3. Login to file server
4. Load TBMI
5. Run WIN.
6. Double click on the DOS icon and load TASKID, followed by NASI. Run NASI in background and full-screen modes. Attempting to use NASI in a window may cause unpredictable results.
7. At the new DOS prompt, enter the command TASKID followed by optional command line parameters in each DOS session. This provides the two way communication needed for virtualizing asynchronous events in the DOS session. TASKID provides a unique ID to TBMI to be used in virtualizing IPX and SPX calls. We recommend that you unload TASKID from the DOS session's memory before closing the session. Do not unload TASKID before task switching.

8. Repeat step #7 for each DOS session you will be using before loading NASI. Load NASI only once.

9. To Exit. Before closing a DOS session with the EXIT command, remember to first type TASKID /U to unload TASKID from that session. Failure to unload TASKID from a session's memory before closing the session may result in loss of data buffers and machine hanging. It is not necessary to unload TBMI after exiting Windows, but you may wish to do so to free up memory.

Limitations

1) You must run in 386 Enhanced mode.

2) Closing the NASI window task without unloading NASI may cause hanging. Therefore, we recommend that you run UNLOAD, followed by TASKID /U.

3) Sometimes Windows needs to be reloaded several times when you have a TSR in place. If, when you load Windows, you are taken back to the DOS prompt, simply reload Windows.

Batch File

If you would like to create an icon to execute a batch file to load NASI, the following sequence of commands is recommended:

TASKID

NASI

(name of communications program)

UNLOAD

TASKID /U

EXIT

Appendix A: NetWare Files

What follows is a listing of all the necessary NetWare files for running Windows 3.0. The listing is grouped according to the ZIP archive that the files are available in, from the Novell NetWire service on the Compuserve Information Service system.

NDD NetWire ZIP Files

The following ZIP files can be obtained from the NDD portion of NetWire. You simply need to type GO NDD at any Compuserve prompt and you will be moved to this location. You can then follow the instructions for downloading any of these files.

DSWIN0.TXT 10318 1-15-91

This file is a text file that describes the various other DSWIN?.ZIP files available from the Novell Download Directory (NDD), and covers some installation tips.

DSWIN1.ZIP 246741 1-15-91

DOC.TXT 36128 5-15-90 11:46a

EMSNET3.EXE 58952 11-27-90 5:26p

EMSNET4.EXE 59432 11-27-90 5:25p

HISTORY.SHL 5489 12-10-90 9:30a

IPX.OBJ 19917 12-18-90 9:48a

LICENSE.TXT 6128 5-22-90 4:14p

NET3.COM 48838 11-27-90 5:25p

NET4.COM 49265 11-27-90 5:24p

README 6023 5-21-90 3:01p

SHELL.TXT 1190 5-22-90 4:12p

XMSNET3.EXE 56056 11-27-90 5:27p

XMSNET4.EXE 56488 11-27-90 5:26p

DSWIN2.ZIP 213457 5-23-90

$RUN.OVL 2400 7-13-89 9:30a

CMPQ$RUN.OVL 2400 7-26-89 10:26p

COMCHECK.EXE 76840 9-01-87 11:53a

COMCHECK.HLP 2673 9-01-87 11:53a

DCONFIG.EXE 22247 6-06-88 11:46a

ECONFIG.EXE 24269 4-14-88 8:21a

IBM$RUN.OVL 2400 7-13-89 9:30a

INT2F.COM 640 7-28-88 11:48a

LICENSE.TXT 6128 5-22-90 4:14p

NLINK.EXE 37633 8-10-89 9:37a

SHCONFIG.EXE 97365 5-04-89 10:57a

SHCONFIG.HLP 33082 5-04-89 10:15a

SHELL.TXT 1190 5-22-90 4:12p

SHELLS.DAT 23 8-17-87 1:44p

SHGEN.EXE 26321 5-04-89 10:06a

SNE1000.LAN 881 5-03-90 9:18a

SNE1000.OBJ 5415 12-27-89 2:30p

SNE2.LAN 133 5-03-90 9:17a

SNE2.OBJ 4781 11-29-89 1:55p

SNE2000.LAN 881 5-03-90 9:18a

SNE2000.OBJ 6121 12-27-89 12:16p

SPCN2.LAN 969 8-15-89 11:22a

SPCN2.OBJ 6003 5-26-88 12:03p

SPS110.LAN 112 8-15-89 11:22a

SPS110.OBJ 6023 8-17-88 4:41p

SRXNET.LAN 1253 8-15-89 11:22a

SRXNET.OBJ 6727 10-10-88 11:43a

STOKEN.LAN 100 8-15-89 11:22a

STOKEN.OBJ 4464 5-05-89 10:11a

SYS$ERR.DAT 6489 7-29-87 9:57a

SYS$HELP.DAT 17343 8-11-87 10:06a

SYS$MSG.DAT 22298 12-22-87 8:42a

VOLUMES.DAT 40 2-10-88 9:31a

DSWIN3.ZIP 430346 5-23-90

AUTOEXEC.BAT 292 5-16-90 8:30a

BINDFIX.EXE 39424 10-30-89 4:29p

FILER.EXE 270781 5-11-90 4:23p

FILER.HLP 65519 8-09-89 4:41p

LICENSE.TXT 6128 5-22-90 4:14p

MAKEUSER.EXE 133595 5-14-90 10:46a

MAKEUSER.HLP 2015 2-22-89 10:03a

NETWARE.PIF 545 4-26-90 1:37p

PRINTDEF.EXE 180211 5-04-90 11:05a

PRINTDEF.HLP 37815 10-27-89 1:32p

SESSION.EXE 111827 8-07-89 9:59a

SESSION.HLP 21066 8-07-89 9:54a

SHELL.TXT 1190 5-22-90 4:12p

DSWIN4.ZIP 347915 5-23-90

CAPTURE.EXE 41025 5-04-90 9:20a

FLAG.EXE 29821 5-09-90 3:54p

FLAGDIR.EXE 27093 3-23-90 11:06a

GRANT.EXE 32385 8-01-89 4:02p

INSTALL.EXE 15061 5-14-90 5:49p

LICENSE.TXT 6128 5-22-90 4:14p

LOGIN.EXE 70301 7-28-89 12:52p

MAP.EXE 45077 8-11-89 4:21p

NCOPY.EXE 58261 8-10-89 4:17p

NDIR.EXE 89226 8-14-89 10:17a

NPRINT.EXE 61021 5-04-90 8:31a

REMOVE.EXE 47674 7-18-89 11:46a

REVOKE.EXE 33549 7-24-89 3:43p

RIGHTS.EXE 17545 7-21-89 2:39p

SHELL.TXT 1190 5-22-90 4:12p

TLIST.EXE 28921 7-18-89 11:57a

VPICDA.386 11063 5-14-90 5:51p

NOVA NetWire ZIP Files

Additionally you may find it necessary to download the following ZIP archive files from the Novell NetWire Forum A. To do this you can simply type in GO NOVA from any Compuserve prompt. You can then follow the forum instructions for downloading these files. These files will be located in the download library 16 Novell New Uploads.

TBMI.ZIP 30401 1-03-91

3270WKST.ZIP 7510 12-18-90 3:20p

IPX.OBJ 19917 12-18-90 9:48a

README.TXT 1007 12-21-90 11:26a

TASKID.COM 2623 12-19-90 3:48p

TBMI.COM 16817 12-19-90 4:34p

TBMI.DOC 9725 11-12-90 8:44p

The following are the unarchived files from the 3270WKST.ZIP archive file included inside the TBMI.ZIP archive file.

3270WKST.ZIP 7510 12-18-90

3270READ.ME 4138 12-18-90 8:18a

JUMP.EXE 11161 12-18-90 8:01a

PTF234.ZIP 36084 1-04-91

IPX.OBJ 19917 12-18-90 9:48a

READ.ME 9472 1-04-91 12:35p

TASKID.COM 2623 12-19-90 3:48p

TBMI.COM 16817 12-19-90 4:34p

UNLOAD.EXE 11349 8-14-89 4:17p

Out-of-Date NetWire ZIP Archives

The following listings of NetWire Windows related ZIP archives that are out of date as of this printing.

DSWIN1.ZIP 242932 5-22-90

This archive contained the version 3.01 Rev. A NetWare shells.

DOC.TXT 36128 5-15-90 11:46a

EMSNET3.EXE 58584 5-08-90 3:40p

EMSNET4.EXE 59000 5-08-90 3:39p

IPX.OBJ 19166 5-07-90 3:18p

LICENSE.TXT 6128 5-22-90 4:14p

NET3.COM 48544 5-08-90 3:37p

NET4.COM 48907 5-08-90 3:35p

NETBIOS.EXE 23088 4-20-90 2:25p

READ.ME 6023 5-21-90 3:01p

SHELL.TXT 1190 5-22-90 4:12p

XMSNET3.EXE 55672 5-08-90 3:42p

XMSNET4.EXE 56056 5-08-90 3:41p

DSWIND.ZIP 201958 9-18-90

This archive contained the version 3.01 Rev. D NetWare shells.

EMSNET3.EXE 58664 9-18-90 2:38p

EMSNET4.EXE 59096 9-18-90 2:42p

NET3.COM 48576 9-18-90 2:21p

NET4.COM 48939 9-18-90 2:22p

REVD.TXT 2207 9-18-90 3:05p

XMSNET3.EXE 55752 9-18-90 2:32p

XMSNET4.EXE 56152 9-18-90 2:41p

SH301E.ZIP 204845 12-20-90

This archive contains the current version 3.01 Rev E NetWare shells, it is just a subset of the files included

in the DSWIN1.ZIP archive.

EMSNET3.EXE 58952 11-27-90 5:26p

EMSNET4.EXE 59432 11-27-90 5:25p

HISTORY.SHL 5489 12-10-90 9:30a

NET3.COM 48838 11-27-90 5:25p

NET4.COM 49265 11-27-90 5:24p

README.TXT 581 12-11-90 4:16p

XMSNET3.EXE 56056 11-27-90 5:27p

XMSNET4.EXE 56488 11-27-90 5:26p

Appendix B: BINDFIX Technical Notes

The following is the Novell Technical bulletins regarding problems with BINDFIX.EXE and the new NetWare shells. Please read this thoroughly before using BINDFIX with the new shells.

Technical Bulletin 255

October 26, 1989

BINDFIX Aberration

When BINDFIX.EXE is executed under the following conditions, data loss will occur:

The NetWare operating system is version 2.15 or below.

The NetWare shell being used on the workstation where BINDFIX.EXE is being executed is a NetWare 386 shell, version 3.0.

The NetWare 386 v3.0 shell is using the SHOW DOTS ON parameter. The default for this parameter is ON.

A user has been deleted from the bindery without the corresponding mail directory being deleted. This occurs when the SYSCON utility being used was an earlier version shipped with NetWare v2.11 or below and the shell being used is a NetWare 386 version 3.0 shell.

The BINDFIX user answers Yes to the question, Delete mail directories of users that no longer exist?

Note: Data loss will occur only when all the above conditions are met.

The Cause

The early versions of SYSCON.EXE were not prepared to deal with the directory entry '.' (from the NET.CFG parameter "SHOW DOTS = ON"). This resulted in the user mail directory not being properly deleted.

BINDFIX.EXE, similarly, is unable to deal with the '.' directory entry. When it attempts to delete mail directories of nonexistent users, it will get caught in a loop of deleting files and directories.

Preventive Measures

We recommend that NetWare users use the 3.0 shell only with NetWare 386 file servers and with NetWare

386 utilities.

In an internetwork environment where it may be necessary to use the 3.0 shell with 2.1x file servers, place the command "SHOW DOTS = OFF" in the NET.CFG file.

Users of some applications need the "SHOW DOTS = ON" parameter set. Make sure that they have insufficient rights to run BINDFIX.EXE.

Make sure that there are no versions of SYSCON.EXE on your internetwork that are earlier than NetWare version 2.12.

Maintain current backups.

Technical Bulletin 256

Date: November 3, 1989

Update to Technical Bulletin 255

The NetWare 386 shell available on NetWire has been changed. The file date on NETX.COM is now 10/31/89. In the earlier version, the default setting for the SHOW DOTS parameter was SHOW DOTS=ON. The default for this parameter has now been changed to SHOW DOTS=OFF. This change is effective only on the shell on NetWire dated 10/31/89. Please be aware that the shell that comes with the NetWare 386 software will still have the default of SHOW DOTS=ON.

The SHOW DOTS default parameter was changed to avoid data loss, as discussed in Technical Bulletin 1-255.

BINDFIX Conclusions

The version of BINDFIX.EXE listed in the ZIP archive below has been modified to handle the '.' when using a 3.0 shell with the SHELL.CFG (or NET.CFG) parameter SHOW DOTS=ON. These circumstances would previously cause BINDFIX to delete all directories and files. Technical bulletins 255 and 256 outline the problem and mention a 3.0 shell which has the default of SHOW DOTS=OFF. This new shell was only a partial fix. This copy of BINDFIX now allows users to comfortably use the 3.0 shell with BINDFIX.

The version of BINDFIX.EXE listed below is available on NetWire in NOVA in the the following ZIP file:

BINDFX.ZIP 23153 12-18-89 12:45p

255.TXT 2344 10-31-89 5:05p

256.TXT 766 11-03-89 8:34a

BINDFIX.EXE 39424 10-30-89 4:29p

READ.ME 480 12-06-89 3:51p

Appendix C: SYSTEM.INI Network Settings

The following SYSTEM.INI settings are ones that can have an affect on network operations either directly, in the case of settings that control things like the network buffer response size that Windows will use, or indirectly, in the case of settings that control how adapter memory (the memory between 640K and 1024K) is used.

The format explanation and definitions used here are from the Windows supplied text files, SYSINI.TXT, SYSINI2.TXT, and SYSINI3.TXT. For further discussions on SYSTEM.INI settings, please consult these files and your Windows User's Manual.

Format

Windows initialization files have the following format:

[section name]

keyname=value

In this example, [section name] is the name of a section. Sections are used to break settings into logical groups. The enclosing brackets ([]) are required, and the left bracket must be in the leftmost column on the screen.

The keyname=value statement defines the value of each setting. A keyname is the name of a setting. It can consist of any combination of letters and digits, and must be followed immediately by an equal sign (=). The value of the setting can be an integer, a Boolean value, a string, or a quoted string, depending on the setting. There are multiple settings in most sections.

You can include comments in initialization files. You must begin each line of comments with a semicolon (;).

The settings will not appear alphabetically in the SYSTEM.INI file. If you want to change a setting, you will have to search for the setting in the appropriate section. Many of the settings explained in this file are rarely needed and will not appear in your SYSTEM.INI file unless you add them yourself.

The syntax, purpose, and recommended method for changing each setting appear in the following format:

SettingName=<value-type>

Default: This is Windows' built-in value for this setting.

Purpose: This paragraph briefly describes the function of the setting.

To change: This sentence states the recommended method for changing the value of this setting.

The <value-type> indicates whether the value should be a number, a letter, a range of numbers, a Boolean value, or something else. If you want to enable a Boolean setting, you can enter: true, yes, on, or 1. If you want the Boolean setting to be disabled, you can enter: false, no, off, or 0.

Many settings listed in this document do not normally appear in your SYSTEM.INI file. Most of these settings have a built-in default value that is present whether or not the setting appears in SYSTEM.INI.

SETUP assigns a value to each setting in the [boot] and [keyboard] sections, and to the Device setting and its synonyms in the [386Enh] section. These settings have no built-in default values. These settings must appear in SYSTEM.INI in order for Windows to function properly, so be careful not to delete them.

[boot] Section

CAUTION: All settings in this section are required. If you modify or delete one of these settings, Windows might not operate properly. There are no built-in default values for these settings; SETUP assigns values based on your system configuration.

The [boot] section contains a list of the drivers and Windows modules that Windows uses to configure itself each time you start it. The following are the [boot] section settings that can affect network operation:

network.drv=<filename>

Default: none

Purpose: Specifies the filename of the network driver you are using.

To change: Choose the Windows SETUP icon from the Main Group window. If you are installing a device driver that is not included with Windows, run SETUP from MS-DOS.

[NonWindowsApp] Section

The [NonWindowsApp] section contains settings that affect the performance of non-Windows applications. The following are the [NonWindowsApp] section settings that can affect network operation:

NetAsynchSwitching=<0-or-1>

Default: 0

Purpose: Indicates whether Windows will allow you to switch away from an application (running in real mode or standard mode) after it has made an asynchronous network BIOS call. The default value of 0 specifies that such task switching is not allowed. Switching away from some applications that make these calls might cause your system to fail. Once Windows detects an asynchronous NetBIOS call, it will not allow switching from the application even if no more of these calls are made. Set this value to 1 if you are sure the applications you use will not receive network messages while you are switched away from them.

To change: Use Notepad to edit the SYSTEM.INI file.

SwapDisk=<drive-colon-directory>

Default: (The directory pointed to by the TEMP environment variable; if there is no TEMP variable, then the default is the Windows directory)

Purpose: Provides the name of the disk drive and directory to which Windows running in real mode or standard mode swaps non-Windows applications.

To change: Use Notepad to edit the SYSTEM.INI file.

[standard] Section

The [standard] section contains settings that are specific to running Windows in standard mode. The following are the [standard] section settings that can affect network operation:

Int28Filter=<number>

Default: 10

Purpose: Specifies the percentage of INT28h interrupts, generated when the system is idle, that are made visible to software that is loaded before Windows. Windows will reflect every nth interrupt, where n is the value of this setting. Increasing this value might improve Windows' performance, but may interfere with some memory- resident software such as a network. Set this value to 0 to prevent INT28h interrupts. But note that setting this value too low will add to system overhead that might interfere with communications applications.

To change: Use Notepad to edit the SYSTEM.INI file.

NetHeapSize=<kilobytes>

Default: 8

Purpose: Specifies the size (in kilobytes) of the buffer pool that standard-mode Windows allocates in conventional memory for transferring data over a network. Some networks require a larger buffer than the default. Increasing this value will diminish the amount of memory available to applications. If no network software is running, this setting will be ignored and no memory will be allocated.

To change: Use Notepad to edit the SYSTEM.INI file.

[386Enh] Section

The [386Enh] section contains information specific to running Windows in 386 enhanced mode, including information used for virtual-memory page swapping. The following are the [386Enh] section settings that can affect network operation:

AllVMsExclusive=<Boolean>

Default: false

Purpose: If enabled, this setting forces all applications to run in exclusive full-screen mode, overriding all contrary settings in the applications' program information files (PIFs). Enabling this setting might prolong the length of the Windows session when you are running network and memory- resident software that is incompatible with Windows.

To change: Use Notepad to edit the SYSTEM.INI file.

Device=<filename-or-*devicename>

Default: none (SETUP assigns appropriate values based on your system configuration.)

Purpose: Specifies which virtual devices are being used with Windows in 386 enhanced mode. This value can appear in two ways: either the name of a specific virtual device file, or an asterisk (*) followed immediately by the device name. The latter case refers to a virtual device that is in the WIN386.EXE file. Synonyms for Device= are Display=, EBIOS=, Keyboard=, Network=, and Mouse=. Filenames usually include the .386 extension. Multiple device lines are required to run Windows in 386 enhanced mode.

To change: Use Notepad to edit the SYSTEM.INI file.

Display=<filename-or-*devicename> (See Device=, above)

Default: none (SETUP assigns an appropriate value based on your system configuration.)

Purpose: Specifies the display device that is being used with Windows in 386 enhanced mode. This setting is a synonym for Device=.

To change: Choose the Windows SETUP icon from the Main Group window.

DualDisplay=<Boolean>

Default: See Purpose.

Purpose: Normally, when running in 386 enhanced mode, the memory between B000:0000 and B7FF:000F will be used by the general system unless a secondary display is detected. If this setting is enabled, this memory will be left unused and available for display adapters. If this setting is disabled, the address range will be available on EGA systems but not under VGA systems, since the VGA display device supports monochrome modes, which use this address space.

To change: Use Notepad to edit the SYSTEM.INI file.

EBIOS=<filename-or-*devicename> (See Device=, above)

Default: none (SETUP assigns an appropriate value based on your system configuration.)

Purpose: Specifies the extended BIOS device that is being used with Windows in 386 enhanced mode. This setting is a synonym for Device=.

To change: Use Notepad to edit the SYSTEM.INI file.

EMMExclude=<paragraph-range>

Default: none

Purpose: Specifies a range of memory that Windows will not scan to find unused address space. This has the side effect of turning off the RAM and ROM search code for the range. The range (two paragraph values separated by a hyphen) must be between A000 and EFFF. This scanning can interfere with some adapters that use the same memory area. The starting value is rounded down and the ending value is rounded up to a multiple of 16K. For example, you could set EMMExclude=C800-CFFF to prevent Windows from scanning the addresses C800:0000 through CFFF:000F. You can specify more than one range by including more than one EMMExclude line.

To change: Use Notepad to edit the SYSTEM.INI file.

EMMInclude=<paragraph-range>

Default: none

Purpose: Specifies a range of memory that Windows will scan for unused address space regardless of what may be there. EMMInclude takes precedence over EMMExclude if you specify ranges that overlap. The range (two values separated by a hyphen) must be between A000 and EFFF. The starting value is rounded down and the ending value is rounded up to a multiple of 16K. For example, you could set EMMInclude=C800-CFFF to ensure that Windows scans the addresses C800:0000 through CFFF:000F. You may specify more than one range by including more than one EMMInclude line.

To change: Use Notepad to edit the SYSTEM.INI file.

EMMPageFrame=<paragraph>

Default: none

Purpose: Specifies the starting paragraph where the 64K page frame will begin when Windows in 386 enhanced mode cannot find a suitable page frame. Allows an EMM page frame in an area containing some unused RAM or ROM. For example, you could set EMMPageFrame=C400 to start the page frame at C400:0000.

To change: Use Notepad to edit the SYSTEM.INI file.

EMMSize=<kilobytes>

Default: 65,536

Purpose: Specifies the total amount of memory to be made available for mapping as expanded memory. The default allocates the maximum possible amount of system memory as expanded memory. You should specify a value for this setting if you run an application that allocates all of the available expanded memory. This will be apparent if, when you run the application, you can never create any new virtual machine. If this value is zero, then no expanded memory will be allocated, but the EMM driver will be loaded. This setting does not prevent the EMM driver from being loaded; use the NoEMMDriver to disable EMM.

To change: Use Notepad to edit the SYSTEM.INI file.

FileSysChange=<Boolean>

Default: on (But in a standard SYSTEM.INI file, SETUP will set FileSysChange=off, disabling this setting.)

Purpose: Indicates whether File Manager will automatically receive messages any time a

non-Windows application creates, renames, or deletes a file. When this setting is disabled, a virtual machine can be run exclusively even when it manipulates files. Enabling this setting can slow down system performance significantly.

To change: Use Notepad to edit the SYSTEM.INI file.

InDOSPolling=<Boolean>

Default: no

Purpose: If enabled, prevents Windows from running other applications when memory-resident software has the InDOS flag set. Enabling this setting is necessary if the memory-resident software needs to be in a critical section to do operations off an INT21 hook. Enabling this setting will slow down system performance slightly.

To change: Use Notepad to edit the SYSTEM.INI file.

INT28Critical=<Boolean>

Default: true

Purpose: Specifies whether a critical section is needed to handle INT28h interrupts used by memory-resident software. Some network virtual devices do internal task switching on INT28h interrupts. These interrupts might hang some network software, indicating the need for an INT28h critical section. If you are not using such software, you might improve Windows' task switching by disabling this setting.

To change: Use Notepad to edit the SYSTEM.INI file.

MapPhysAddress=<range>

Default: none

Purpose: Specifies the address range (in megabytes) in which the memory manager will preallocate physical page-table entries and linear address space. Use this setting if you are using a DOS device driver (such as an older version of RAMDrive that uses extended memory) that needs this contiguous memory.

To change: Use Notepad to edit the SYSTEM.INI file.

MaxPagingFileSize=<kilobytes>

Default: none

Purpose: Specifies the maximum size (in kilobytes) for a temporary swap file.

To change: Use Notepad to edit the SYSTEM.INI file.

MinTimeSlice=<milliseconds>

Default: 20

Purpose: Specifies the minimum amount of time (in milliseconds) a virtual machine will be allowed to run before other virtual machines can take over. A smaller value (such as 10 milliseconds) will make multitasking appear smoother, but will diminish the overall system performance.

To change: Choose the 386 Enhanced icon from the Control Panel window.

MinUserDiskSpace=<kilobytes>

Default: 500

Purpose: Tells Windows how much disk space (in kilobytes) to leave free when creating a temporary swap file. You would want to use this setting if your system's paging drive has less available space than Windows can use for paging. This setting has no effect if a permanent swap file exists.

To change: Use Notepad to edit the SYSTEM.INI file.

NetAsynchFallback=<Boolean>

Default: false

Purpose: If enabled, tells Windows to attempt to save a failing NetBIOS request. When an application issues an asynchronous NetBIOS request, Windows will attempt to allocate space in its global network buffer to receive the data. If there is insufficient space in the global buffer, Windows will normally fail the NetBIOS request. If this setting is enabled, Windows will attempt to save such a request by allocating a buffer in local memory and preventing any other virtual machines from running until the data is received and the timeout period (specified by the NetAsynchTimeout setting) expires.

To change: Use Notepad to edit the SYSTEM.INI file.

NetAsynchTimeout=<seconds>

Default: 5.0

Purpose: Specifies the timeout period (in seconds) when Windows needs to enter a critical section in order to service an asynchronous NetBIOS request. It is used only when NetAsynchFallback is enabled. This value can include a decimal (such as 0.5).

To change: Use Notepad to edit the SYSTEM.INI file.

NetDMASize=<kilobytes>

Default: 32 on Micro Channel (TM) machines 0 on non-Micro Channel machines

Purpose: Specifies the DMA buffer size (in kilobytes) for NetBIOS transport software if a network has been installed. In this case, the buffer size is the larger value between this value and the value of DMABufferSize.

To change: Use Notepad to edit the SYSTEM.INI file.

NetHeapSize=<kilobytes>

Default: 12

Purpose: Specifies the size (in kilobytes) of the buffers that Windows in 386 enhanced mode allocates in conventional memory for transferring data over a network. All values are rounded up to the nearest 4K.

To change: Use Notepad to edit the SYSTEM.INI file.

Network=<filename-or-*devicename> (See Device=, above)

Default: none (SETUP assigns an appropriate value based on your system configuration.)

Purpose: Specifies the type of network you are using with Windows in 386 enhanced mode. This setting is a synonym for Device=.

To change: Choose the Windows SETUP icon from the Main Group window.

Paging=<Boolean>

Default: yes

Purpose: Enables or disables demand paging (virtual memory). You would disable this setting only if you need the disk space normally used for a temporary swap file.

To change: Use Notepad to edit the SYSTEM.INI file.

PagingDrive=<drive-letter>

Default: none

Purpose: Specifies the disk drive where Windows in 386 enhanced mode will allocate a temporary swap file. This setting is ignored if you have a permanent swap file. If you don't have a permanent swap file and no drive is specified or the specified drive does not exist, Windows will attempt to put your temporary swap file on the drive containing your SYSTEM.INI file. If the specified drive is full, paging will be disabled.

To change: Use Notepad to edit the SYSTEM.INI file.

PSPIncrement=<number>

Default: 2

Purpose: Specifies the amount of additional memory, in 16-byte increments, that Windows should reserve in each successive virtual machine when the UniqueDOSPSP setting is enabled. The setting that will work best for your machine might vary depending on your memory configuration and the applications you are running. Valid values are 2 through 64. See UniqueDosPSP for more information.

To change: Use Notepad to edit the SYSTEM.INI file.

ReflectDosInt2A=<Boolean>

Default: false

Purpose: Indicates whether Windows should consume or reflect DOS INT 2A signals. The default means Windows will consume these signals and therefore run more efficiently. Enable this setting if you are running memory-resident software that relies on detecting INT2A messages.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMEMSLimit=<kilobytes>

Default: 2048

Purpose: Specifies how many kilobytes of expanded memory Windows should be permitted to use. Setting this value to 0 prevents Windows from gaining access to any expanded memory. Setting it to #1 gives Windows all the available expanded memory that it requests.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMEMSLocked=<Boolean>

Default: no

Purpose: Indicates whether to swap Windows' expanded memory to the hard disk. Locking expanded memory can improve the performance of a Windows application that uses it, but locking it slows down the rest of the system.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMEMSRequired=<kilobytes>

Default: 0

Purpose: Specifies how many kilobytes of expanded memory must be free in order to start Windows. Leave this setting at zero if no Windows application requires expanded memory.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMV86Locked=<Boolean>

Default: false

Purpose: If enabled, causes the virtual-mode memory being used in the system virtual machine to remain locked in memory rather that being swapable out to disk. Because Windows handles this process, there is no known reason to enable this setting.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMXMSLimit=<kilobytes>

Default: 2048

Purpose: Specifies the maximum amount of memory (in kilobytes) the extended memory driver will allocate to DOS device drivers and memory-resident software in the system virtual machine. Set the value to #1 to give an application all the available extended memory that it requests.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMXMSLocked=<Boolean>

Default: no

Purpose: Indicates whether to swap the memory allocated by the extended memory driver to the hard disk. Locking the XMS memory (enabling this setting) can improve an application's performance, but it slows down the rest of the system.

To change: Use Notepad to edit the SYSTEM.INI file.

SysVMXMSRequired=<kilobytes>

Default: 0

Purpose: Specifies how many kilobytes of extended memory must be reserved by the XMS driver in order to start Windows. Leave this setting at zero if there are no XMS users in the system virtual machine.

To change: Use Notepad to edit the SYSTEM.INI file.

TimerCriticalSection=<milliseconds>

Default: 0

Purpose: Instructs Windows to go into a critical section around all timer interrupt code, and specifies a timeout period (in milliseconds). Specifying a positive value will assure that only one virtual machine at a time will receive timer interrupts. Some networks and other global memory-resident software may fail unless this setting is used. However, using it will slow down performance and can make the system sluggish or seem to stop for short periods of time.

To change: Use Notepad to edit the SYSTEM.INI file.

TokenRingSearch=<Boolean>

Default: true

Purpose: Tells Windows whether to search for a token ring network adapter on machines with IBM PC/AT (R) architecture. Disable this setting if you are not using a token ring card and the search interferes with another device.

To change: Use Notepad to edit the SYSTEM.INI file.

UniqueDOSPSP=<Boolean>

Default: false (see below for exception)

Purpose: If enabled, tells Windows to start every application at a unique address (PSP). Each time Windows creates a new virtual machine to start a new application, Windows reserves a unique amount of memory (i bytes) below the application. For example, the first application would be loaded at address M, the second at address M+i, the third at M+2i, and so forth. The amount of memory i is determined by the PSPIncrement setting (earlier in this section). These settings should help assure that applications in different virtual machines all start at different addresses. Some networks use applications' load addresses to identify the different processes using the network. On such networks, failing to enable this setting might cause one application to fail when you exit another, because the network interprets them as the same. However, enabling this setting will leave slightly less memory for non-Windows applications. If you are running a network based on Microsoft Network or LAN Manager, the default value is true. See the NETWORKS.TXT online document to find out whether the network you are running is one of these.

To change: Use Notepad to edit the SYSTEM.INI file.

VirtualHDIrq=<Boolean>

Default: on

Purpose: Allows Windows in 386 enhanced mode to terminate interrupts from the hard disk controller, bypassing the ROM routine that handles these interrupts. Some hard drives might require that this setting be disabled in order for interrupts to be processed correctly. If this setting is disabled, the ROM routine handles the interrupts, which slows the system's performance.

To change: Use Notepad to edit the SYSTEM.INI file.

WindowKBRequired=<kilobytes>

Default: 256

Purpose: Specifies how much conventional memory (in kilobytes) must be free in order to start Windows.

To change: Use Notepad to edit the SYSTEM.INI file.

WindowMemSize=<number-or-kilobytes>

Default: #1

Purpose: Limits the amount of conventional memory Windows can use for itself. The default value (#1) indicates that Windows can use as much of this space as it needs. You can try entering a positive value less than 640 if there is not enough memory to run Windows in 386 enhanced mode.

To change: Use Notepad to edit the SYSTEM.INI file.

Appendix D: BibliographyBooks

Microsoft Windows User's Guide Version 3.0. Microsoft Corporation. 1985-1990.

Microsoft Windows User's Guide Version 2.0. Microsoft Corporation. 1987.

Using Microsoft Windows/386. Microsoft Corporation. 1987.

Microsoft Windows User's Guide Version 2.0. Microsoft Corporation. 1987.

Graphic Facts About Networking with Windows. Automated Design Systems, Inc. 1988-1990.

Saber LAN Setup Guide for Windows Version 1.0. Saber Software Corporation 1990.

Duncan, Ray. Extending DOS. Reading, MA.: Addison-Wesley Publishing Co., Inc. 1990.

Norton, Peter, and Yao, Paul. Peter Norton's Windows 3.0 Power Programming Techniques. New York, NY.: Bantam Books, Inc. 1990.

Articles

Geary, Michael. An Introduction to Microsoft Windows Version 3.0: A Developer's Viewpoint. Microsoft Systems Journal. (July 1990): 1- 28.

Electronic Documents

WINDOWS 3.00 AND NETWORKS. Microsoft Corporation. 1990.

NETWORKS.TXT. Microsoft Corporation. 1990.

SYSINI.TXT, SYSINI2.TXT, SYSINI3.TXT. Microsoft Corporation. 1990.

WININI.TXT, WININI2.TXT. Microsoft Corporation. 1990.