Q: What is VxD?
A: A VxD, or Virtual Device Driver, is a 32-bit multiplexing device driver that manages data exchanges between Windows applications and system services. In the context of TCP/IP networking, a TCP/IP stack in a VxD accepts requests for network services from applications, properly formats those requests according to the TCP/IP protocol specifications, sends them to the network hardware device drivers and subsequently handles any response or responses, returning the requested data to the application that made the request. VxD is the most efficient and least expensive utilization of the CPU, as TCP/IP-related activity is done at Ring 0 without going to Ring 3 (DLL) or DOS Virtual Machine (TSR), ensuring the best performance.

Q: What benefit does a VxD deliver?
A: A VxD delivers a number of benefits. A VxD works the same way as the operating system itself, delivering the sampotential degree of service, in terms of available memory, speed of operation and number of applications it can support. In practical terms, a VxD makes the maximum amount of memory available for other programs, particularly those that run in a DOS box. That means people can do more with the Windows workstation. Those applications that use the VxD services should run faster, because the VxD has the ability to control its own operation more effectively, in cooperation with the operating system. People can also get more work done by using as many network applications (or applications that make use of the network) as they need at any one time. Those applications can be true Windows applications or applications that run in a DOS box. A VxD takes advantage of the full 32-bit architecture of future Windows versions, such as Chicago, now in beta. This allows for greater performance and greater service, opening up the possibility of moving a greater range of information, including sound and pictures, at speeds acceptable to people using the machine. In short, a VxD implementation of the TCP/IP networking protocols turns the Windows machine into a more powerful, more capable workstation.

Q: Why use a VxD and not a DLL?
A: A VxD provides immediate response without any penalty. A Dynamic Link Library (DLL) only becomes active when some application calls the DLL and uses it. As soon as all applications using the DLL quit, Windows unloads the DLL from memory, removing the services the DLL provides. This means, in the context of networking, that the DLL-based workstation no longer "hears" the network - no longer receiving data communications. This can make managing a network with many PCs mor edifficult, since not all PCs can be monitored constantly. A VxD, because it is always there, makes the management of a network easier, allowing workstations to handle network traffic as it occurs. A VxD can easily service applications running in a DOS box. A DLL cannot provide service to applications running in a DOS box, which may preclude the use of existing applications. Windows "fixes" a DLL that services interrupts, such as those generated by a network interface card, in real mode DOS memory below 1 MB while it is running. Depending on the size of the DLL, this can severely limit the use of other programs, particularly those that run in a DOS box in Windows. A VxD has no such mechanism or limitation.

Q: Why use a VxD and not a TSR?
A: Primarily because a TSR (terminate-and-stay-resident) program occupies real-mode DOS memory, limiting the use of other programs, including Windows itself if the TSR is big enough. Most often, the impact of a TSR is felt when the user attempts to use one or more programs in a DOS box in Windows and finds the machine does not have enough memory to run the program. A VxD needs little or no real-mode DOS memory. The use of a TSR also requires a method for communicating data between the TSR, which loads into memory and becomes active before Windows starts, and any Windows programs. This mechanism necessarily incurs a reduction in performance, though this can be minimized. This TSR-to-Windows mechanism also introduces a complexity that can be a critical point of failure in the system. A VxD is fully integrated into Windows, eliminating the need for this communication mechanism. The makers of Windows, Microsoft, has informed the general marketplace that future versions of Windows (such as Windows 4.0 or Chicago, as it as known) will not support TSRs. Whether this proves completely true or not, Microsoft has made its intentions and directions clear. Chicago will not require the presence of DOS before starting Windows, thereby changing the native environment of TSR programs in ways that cannot be predicted now.

Q: Is a VxD somehow more complicated, harder to use, more difficult to install?
A: A VxD is no more difficult to install than any other Windows system-level software. Once installed, using the same sort of Windows installation program as a word processing or spreadsheet application, a VxD simply loads and goes to work when Windows starts up, requiring no action on the part of the user. A good VxD implementation will not then introduce any noticeable effect on any Windows program, other than to provide networking service as promised. Technically speaking, a VxD is complex and powerful; a poorly implemented VxD can easily crash the Windows machine.

Q: Is there a way to distinguish better VxD implementations from others?
A: Yes, in much the same way you distinguish better applications of any sort. Does the implementation lock up or crash the machine? Does the implementation require a lot of system resources, such as memory or time to run? Does it perform well under stress? (You can tell by moving the mouse, and response of the application to the user input.) How much load can it take? How many applications can it service at once before the user notices a slowdown in response? Does it perform the required or desired tasks? In the case of TCP/IP networking systems, you can also look at the origins of the VxD implementation. Up until now, with the impending release of a fully 32-bit Windows system, most of the TCP/IP networking systems available for Windows originated in DOS, in a 16-bit architecture. Since Windows itself was and is an extension of

DOS, requiring the presence of DOS before Windows can even start, many of the existing TCP/IP networking products migrated to Windows from DOS, similarly retaining the 16-bit architecture. Some even kept the DOS-based TSR method for implementing basic transport service. In the next release of Windows, DOS is no longer required, and Windows moves to a 32-bit architecture. This makes Windows much more similar, as an operating environment, to traditional multitasking, multiuser operating systems such as UNIX or VMS. The TCP/IP protocols were originally developed in such 32-bit, preemptive operating system environments, and many excellent implementations are available in that environment. With Windows moving to 32 bits and a preemptive operating system environment, it becomes possible to move TCP/IP from UNIX or VMS to Windows without significant reworking, taking full advantage of the experience and field testing of TCP/IP in those environments. A TCP/IP implementation moved from a 32-bitoperating system to a 32-bit VxD implementation may prove to deliver greater, more stable service, in terms of the product and in terms of the support delivered by the vendor.

Q: Does a VxD require new or special network applications?
A: Yes and no. A VxD transport system will not require any new or special applications, so long as the vendor of that VxD hastaken care to make existing applications work. The likelihood of this is very high (even Microsoft is doing it). The likelihood of VxD transport vendors continuing to support Windows Sockets is also very high, allowing the increasinglylarge number of existing Windows Sockets applications to run unaltered. To take full advantage of the next version of Windows (Chicago, now in beta), a new class of applications will emerge. Theseapplications will take full advantage of the 32-bit architecture of Windows applications, as opposed to the existing 16-bitarchitecture. Applications built to the 32-bit architecture can in turn offer more service, both to the user and also to and fromthe network, because the VxD transport system is 32-bits. A 32-bit architecture makes available more memory space, lets agreater number of applications run at the same time, and provides a faster path for data to travel from the network to theapplication, and vice versa. This is similar to putting a bigger engine in your car. The car might not go faster; however, the greater horsepower makes it easy to add such services as air conditioning and poweraccessories. Network applications can provide similar additional services, for the comfort and protection of the user.

Q: Does VxD support Windows Sockets?
A: Yes, in much the same way Windows Sockets applications work now. The Windows Sockets DLL will still make datacommunications between an application and the underlying transport system possible.Nothing, except perhaps make it easier for those applications to gain access to the network. In a sense, a VxD gets out of theway of other applications, because it will not occupy memory space that other applications want to use or need to use. Such applications should run more smoothly. There is one important note about the Windows system to make here. The core of Windows (the Virtual Machine Manager, orVMM) is implemented in a 32-bit architecture such as Windows versions 3.1 and Chicago. However, applications are still only16-bits, and these applications must still cooperate with other applications by yielding control of the operating system whenever possible, allowing the VMM to run other active applications, including those running in the background. This need to yieldproved difficult to implement in Windows networking applications. Users felt the impact when non-network applications ranslowly or not at all while a network application was active. In the next generation of Windows, applications can move to 32-bitarchitectures. In so doing, those applications can take advantage of the VMM's ability to pre-emptively multitask, meaning the32-bit applications do not need to yield whenever possible. Rather, the Windows VMM will manage applications, ensuring thateach runs as needed, when needed. Windows applications still using a 16-bit architecture will still need to yield, as always. Thisis one way in which Windows will maintain backward compatibility, meaning existing applications will still run unaltered on thenext version of Windows. Network applications built in the 16-bit architecture must also yield, and therefore continue to run therisk of interfering with other applications, such as word processing or spreadsheet. Future network applications built in the32-bit architecture can take advantage of the pre-emptive nature of future (Chicago, now in beta) Windows systems, providinga much greater degree of compatibility and cooperation with other Windows applications. Look for 32-bit networkapplications.

Q: Is VxD technology new?
A: No. VxDs were possible in Windows 3.0 and are possible in Windows 3.1 and later, including Windows for Workgroupsversion 3.11. VxDs have been around for four or five years now.

Q: Technically speaking, what is a VxD?
A: A VxD runs at ring 0 as part of the Windows 386 virtual machine manager (VMM), in 32-bit flat mode. Ring 0 operationgives the VxD operating-system level of protection and the CPU allows the VxD to execute any 386 instruction. Ring 0 operation also allows the VxD to restrict access to memory addresses or I/O ports by higher ring level applications, thusallowing the VxD to process any exceptions. A VxD is the only program with unobstructed access to the hardware. Thus, if a TCP/IP VxD owns the hardware interrupt forthe network interface card (or driver), the VxD receives the IRQ directly from the Virtual Programmable Interrupt Controller(VPICD)) without ring transitions. A VxD is also given priority on all actions in a VM. A VxD has control of the interrupts, I/O ports and memory. In short, aVxD can do a lot, and do it very fast.