Microsoft®
Windows™ “Chicago”
Reviewer’s Guide

Beta-1

The information discussed in this guide is based on features and functionality present either in the Beta-1 release of Chicago, or planned for a future release. The discussion of Chicago herein, does not represent a commitment on the part of Microsoft for providing or shipping the features and functionality discussed in the final retail product offerings of Chicago.

Table of Contents

Introduction

Welcome

As the successor to Microsoft® MS-DOS®, Windows™ 3.1 and Windows for Workgroups 3.11, Windows "Chicago" is the next major release of the standard operating system for the desktop and portable PC. There is something for everyone, whether it's a more intuitive way to work, new capabilities like "surfing the information highway," or better support for managing a 1,000 PC installation site.

This *Reviewer's Guide* describes and explains the features in Microsoft's Windows "Chicago" beta. Since the size and scope of the Chicago project is quite large, the Introduction chapter contains a brief summary of the key features and customer benefits. This chapter also provides some context on the principles guiding the Chicago product's development and our vision of its role with customers and the industry.

Chicago Mission

Where We've Been

Over the past decade, the PC industry has delivered innovative, cost-effective products that have made the personal computer a widely-used tool both in the office and at home. These products were enabled by a number of key advances during the 1980s and early 1990s:

- The adoption of MS-DOS as an operating system standard for PCs, providing a platform for application development
- Rapid decreases in price-performance ratios due to innovations in PC components, especially with the introduction of Intel i386™ and i486™ microprocessors
- The wide adoption of the Microsoft Windows operating system for PCs, which made PCs much easier to use because of its appealing graphical user interface, multitasking capabilities, and the new generation of graphical applications that Windows made possible

Where We Are Today

Although the PC has made dramatic gains over the past decade, a number of limitations keep current users taking full advantage of their PCs and discourage others from beginning to use them:

- **PCs are still not easy enough.** Many people perceive PCs as hard to use. Tasks such as connecting to a printer or adding a CD-ROM drive are too complicated for the typical computer user. Just learning how to use the PC is difficult because the PC does not lend itself to learning through experimentation. The wrong configuration or set of keystrokes can, for example, inexplicably cause programs to disappear. Many users are frustrated learning even basic tasks like launching or switching between applications.
- **Users want software that takes full advantage of the power of their hardware.** The performance delivered by hardware—the CPU, video system, CD-ROM drive, video subsystem, and so on—is constantly accelerating. And new hardware—such as wireless, PCMCIA, and MPEG—continuously arrives on the market. Users want their software to exploit all this power to its fullest.
- **Connecting to a network is hard.** The two primary PC operating systems, MS-DOS and Windows, were not designed with the explicit goal of easy connectivity to a network. Consequently, notwithstanding the prevalence of networks in organizations of all sizes, the basic task of connecting a PC to a server and to other PCs is still a challenge today.
- **PCs are expensive to support.** The cost of PC hardware is small relative to the costs of installing, configuring, and maintaining the PC and of training and supporting the PC user. These costs must be reduced to make PCs a more cost-effective tool for business.

Where We're Headed

Windows 3.1 moved the PC platform forward by making PCs easier to use. Yet today's customer problems highlight the need to significantly further the ease, power, and overall usefulness of the PC. Windows Chicago goes beyond simple ease of use. It not only enables a new range of people to become PC users by making the PC dramatically easier to use, but also enables a wide new range of uses for the PC for existing users as follows:

- **Chicago makes PCs even easier to use.** Windows 3.1 put a friendly interface on top of MS-DOS to make common PC tasks easier. In Chicago, the goal is to make those tasks more intuitive, or where possible, automatic. The addition and configuration of new hardware devices on the PC is one example. Windows Chicago automatically loads the appropriate drivers, sets IRQs, and notifies applications of the new capabilities of the hardware device without any action by the user. A redesigned user interface, highlighted by the Windows Taskbar, makes computing more automatic for novices—Chicago usability tests show a ten-fold improvement over Windows 3.1 in time to complete certain common tasks such as starting an application—and makes the power of the PC more discoverable for intermediate and advanced users.

- **Chicago is a faster and more powerful operating system.** Ease on the surface requires power and speed at the core, and Chicago's modern, 32-bit architecture meets these requirements. Freed from the limitations of MS-DOS, Chicago preemptively multitasks for better PC responsiveness—so users will no longer have to wait while the system copies files, for example—and also delivers increased robustness and protection for applications. Chicago also provides the foundation for a new generation of easier, more powerful multi-threaded 32-bit applications. And most importantly, Chicago delivers this power and robustness on today's average PC platform while scaling well to take advantage of additional memory and CPU cycles.
- **Chicago integrates network connectivity and manageability.** Windows 3.1 gave end-users the power to better use their PCs, but it did not make the same strides for MIS organizations. Windows Chicago addresses this deficiency by providing a system architecture that makes basic network connectivity easy by integrating high performance, 32-bit client support into the operating system—including a 32-bit client for Novell® NetWare®—and goes beyond simple connectivity by enabling the central management and control of the PC. Chicago's user profiles, policies, and ability to pass through server-based security make it much easier for MIS organizations to administer and support large numbers of PCs within the corporation.

  Chicago is more than the next generation of Windows—it is a catalyst that will move the PC industry to a higher level of usefulness for end-users. We expect the release of Chicago to spawn not only a new generation of PCs and peripherals that support Plug and Play, but a new generation of powerful, 32-bit Windows applications as well.

How We Get There

In many ways, Chicago's development has been guided by the primary principle of making all of the great technological improvements translate to practical benefits for users, and making these benefits easily and inexpensively available to everyone. This means several things:

- **Compatibility with existing MS-DOS and Windows-based applications.** Chicago is designed to add significant value to any PC without requiring additional software or hardware, and compatibility with existing applications is a must. Chicago not only supports the enormous range of existing MS-DOS and Windows applications, but goes a step beyond by fixing key compatibility deficiencies of Windows 3.1. It provides, for example, much better support for demanding MS-DOS–based applications. With Chicago, you can run your cool games right from Windows.
- **Compatibility with existing hardware.** The wide range of available hardware for Windows 3.1—from scanners to plotters to fax modems to video capture boards—is enormously valuable to users. While providing support for new,

easier-to-use peripherals through Plug and Play, Chicago also preserves the huge investment both users and manufacturers have by maintaining compatibility with existing peripherals and their associated device drivers.

- **Equal or better performance than Windows 3.1.** Keeping with the goal of adding value to existing PCs without needing extra hardware and software, Chicago matches or exceeds the performance of Windows 3.1 on today's average PC (386DX with 4MB of RAM). And as more memory is added to the PC, performance scales faster than Windows 3.1. For customers this means access to all of Chicago's new features should not cost them any performance penalty.
- **Safe, hassle-free upgrade and migration.** Chicago makes the upgrade process as easy as possible with a Setup program that upgrades cleanly over Windows 3.*x*. Chicago also includes the Windows 3.1 File Manager and Program Manager so that end-users can migrate to the new user interface at their own pace.

A Quick Preview of Chicago's Top Features

This section briefly outlines some of the key new features in Chicago and the problems they solve for customers. Since the scope of new features is broad and their appeal wide, we've organized them into features for end-users and MIS organizations in terms of how they provide the following benefits relative to Windows 3.1:

- Even easier
- Faster and more powerful
- Compatible

Even Easier

For End-Users

For end-users and MIS organizations alike, improving ease of use goes beyond fixing problems with Windows itself; the improvements in Chicago also encompass the hardware, connectivity, and applications as well. Less-experienced users find the plethora of overlapping Windows, minimizing and maximizing too complex to easily master. More experienced users crave greater efficiency. Chicago offers these solutions:

- **New user interface.** A blizzard of improvements greatly enhance learnability, usability, and efficiency for users of all levels of expertise. New users can get started more quickly, and experienced users can fully unlock the power of their PCs.

- **Plug and Play.** The goal of Plug and Play is simple: When a user installs a new hardware device, it just works.
- **Long Filenames.** The end of 8.3, long filenames are an important example of Chicago's many usability improvements.

For MIS Users

MIS organizations increasingly find their job more difficult as the number of PCs in a given corporation increases more rapidly than their support staff. In addition, the introduction of Windows 3.1 led to a series of security, reliability, and management issues because of its lack of integrated connectivity and lack of infrastructure for being managed. Some features that attack these problems are:

- **Built-in networking.** Whether you're running NetWare or Microsoft Networks; IPX/SPX, TCP/IP or NetBEUI; NDIS or ODI, Chicago has native, integrated support for your network. And additional LANs are easily supported.
- **The Registry.** By holding all pertinent information about the system—installed hardware, installed software, user preferences and rights—and by exposing its contents remotely through a wide variety of industry standard interfaces —SNMP, DMI, Win32® plus RPC—the Registry provides the foundation for a highly-manageable PC.
- **User profiles**. Chicago has the ability to expose different functionality depending on who has logged into the PC. A network administrator could, for example, prevent Joe from deleting program items no matter what PC he logs in on, or enforce a policy that everyone in a particular group have passwords of at least six characters.
- **Pass-through user- level security**. A Chicago PC can require that a remote user and their password be passed-through to and validated by a NetWare server or Windows NT™ Advanced Server before allowing the user to do something—for example, change the PC's Registry or access its files and printers over the network or remotely.
- **Network backup agents**. Chicago includes agents that support industry-leading server backup products.

Faster and More Powerful

For End Users

Another major area of concern for end-users is improving the efficiency and power with which they use Windows. Users care about getting their work done faster. They'd like to run more than one application or computer process at a time so they spend less time waiting on their PC. They'd like to be more effective without sacrificing system stability or performance. And perhaps most important of all, they'd like to escape the feeling that they take advantage of only a small fraction of what their PC can do.

Chicago is designed to anticipate and exploit key emerging trends and technologies. The need for seamless mobile computing, for example, is more important than ever as more hardware power is packed into smaller and lighter designs, and more users work at home or on the road. The explosion of the home market means that users demand more powerful multimedia applications for their computers, and this means better multimedia support from the operating system.

Key features that bring more power and speed to users include:

- **True preemptive multitasking**. Chicago can preemptively multitask 32-bit applications smoothly and efficiently.
- **Scaleable performance**. Chicago performance increases more rapidly than Windows 3.1 as the amount of RAM increases because of its high-performance, 32-bit architecture.
- **Support for 32-bit applications.** Chicago's support for the Win32 API means that users can look forward to a new generation easier, more powerful multi-threaded 32-bit applications.
- **Increased robustness**. New features mean greater robustness and protection for existing MS-DOS and Windows applications, and the highest level of protection for new 32-bit Windows applications.
- **Mobile computing anywhere**. Chicago provides both a remote networking client that allows dial-in to any network, including the Internet, running IPX/SPX, TCP/IP or NetBEUI over PPP, as well as a dial-in server that lets any Chicago PC act as a secure, single-line dial-in gateway to a network.
- **Faster printing.** Chicago's new 32-bit printing subsystem means a lot less time waiting for print jobs to spool and to finish.
- **High-performance multimedia components**. Both the video playback engine (Video for Windows) and CD-ROM file system (CDFS) are new 32-bit components that deliver smoother video and sound reproduction.
- **More memory for MS-DOS applications**. Chicago's use of protected-mode device drivers and file systems means users will routinely get 600K+ free conventional memory in each MS-DOS session even while connected to the network and using a CD-ROM drive and using a mouse and so on.

For MIS Users

MS-DOS and Windows were not architected to work in a networked environment. The result is a string of well-known problems to MIS professionals, including not having enough real-mode memory to run the MS-DOS-based mission critical application for the business, instability under Windows (for example, turning off a Novell® NetWare server completely hangs all connected Windows machines.

Just some of the Chicago product features that solve these problems for MIS organizations are:

- **32-bit NetWare connectivity**. Chicago includes a 32-bit network client NetWare that is fast, reliable and requires zero-footprint in conventional memory. Chicago also includes a similar client for Microsoft Networks.
- **Multiple network support**. A Chicago PC can have multiple network clients (NetWare, Windows NT Advanced Server, and so on) and multiple transports (including IPX/SPX, TCP/IP, NetBEUI) loaded and running simultaneously.
- **Universal information client**. Chicago's information client (code-named Capone) is integrated seamlessly into the user interface, and allows a wide range of services—whether email, online services, group applications, and others—to plug in. This gives the user a single interface onto a world of information, and the MIS administrator a single client that supports multiple email and other systems. Chicago includes support for two services: Microsoft Mail and faxing (Microsoft At Work™ fax software).

Compatible

For End Users

If an operating system upgrade requires new software, more memory, or new hardware, then the upgrade's cost is far higher than just its purchase price. Unfortunately, users usually need to wait a substantial amount of time—usually until their next PC purchase—before benefiting from the latest technology.

One of Chicago's biggest goals was to allow everyone to benefit from the latest version of Windows and thus the following features:

- **Compatibility with existing MS-DOS and Windows-based applications**. Chicago works with and improves the software users have today.
- **Same or better performance**. As long as a user has at least a 386DX with 4MB RAM, they are assured that Chicago will run their system at least as fast as Windows 3.1, and in many cases, faster. Chicago requires no additional RAM to maintain performance.
- **Backwards compatibility with existing hardware devices**. Chicago supports existing hardware and device drivers while also enabling next generation, easier-to-use hardware through Plug and Play.

For MIS users

MIS organizations have had similar problems as those of end-users in terms of the cost to upgrade to a new operating system. Compatibility with today's hardware and software is even more important, not only because of the larger scale of the upgrade, but because of compatibility problems with having differing operating system platforms within the organization. Having to support more than one platform only multiplies the problems MIS organizations face. MIS professionals worry about re-training users and about the need to migrate users to a new platform quickly, easily, and in an orderly way.

Chicago addresses these needs through:

- **Network compatibility**. In addition to its 32-bit client for Novell NetWare, Chicago is compatible with existing real-mode network clients and existing logon procedures like login scripts used by NetWare.
- **Inclusion of the Windows 3.1 Program and File Managers**. An explicit Chicago goal is to allow users and their MIS departments to change to the new user interface at their own pace.

Summary

The mission for Windows Chicago is to go beyond making PCs easier to make them truly usable. This means a more intuitive and automatic PC that also integrates the latest technologies and offers superior responsiveness and stability. For end-users, this means an even easier, faster, and more powerful PC that is compatible with today's existing software and hardware. And Chicago aims to make the upgrade and transition easy, without pain and without loss of performance or capability.

The size and scope of the Chicago product is awesome and you'll find the features in the first beta are worthy of the mission. The rest of this guide goes into more detail on the specific benefits of the many Chicago feature improvements and innovations. Enjoy.

Chicago's Place in the Microsoft Windows Operating System Product Line

Microsoft's operating system product line provides customers and developers with a rich set of services which take full advantage of the broad range of hardware platforms available today, from small form-factor portable systems to multiprocessor servers. The members of the Windows operating system family all share a consistent user interface and a consistent programming environment to make the entire product line easier to learn and use for both end-users and developers.

Which Products Are Available Today?

Today the Windows operating system product line includes Microsoft Windows operating system version 3.1 and Microsoft Windows for Workgroups 3.11, which are designed to run personal and business productivity applications on mainstream PC platforms. The most recent additions to the product line are Microsoft Windows NT Workstation and Windows NT Server operating systems, which are designed to run the most demanding applications on high-end workstation and server platforms.

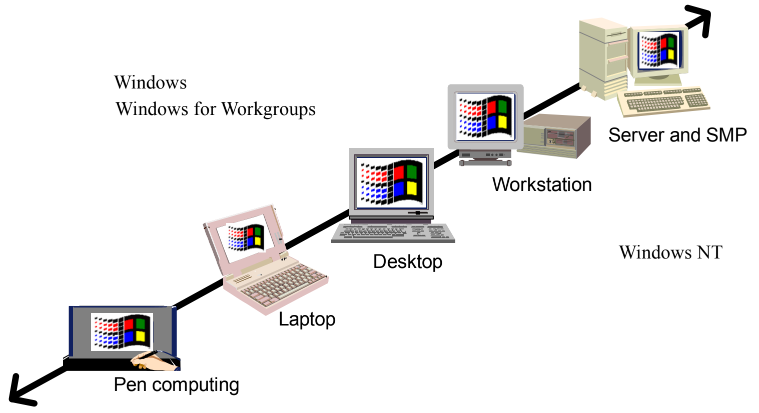


Figure 1. Consistent Platform for Development, Deployment, and Training

Which Products Will Be Available in the Future?

Microsoft continues to update and enhance the Windows operating system product line to keep pace with the growing and changing user needs. Chicago is the code name for the development project to create the successor to Windows 3.1 and Windows for Workgroups, and moves Windows a major step forward. Chicago provides an easier to use system with increased functionality for mainstream desktop platforms while leveraging the investment customers have in their existing applications and systems.

The successor to Windows NT is the project code named Cairo, which will add directory services, an object file system, advanced querying capabilities, and an object-oriented development model. Daytona is an interim release of Windows NT focused on size and performance improvements plus enhanced connectivity services. When a new Windows platform ships, other members of the Windows family will be updated to maintain consistent user interfaces and application programming interfaces.

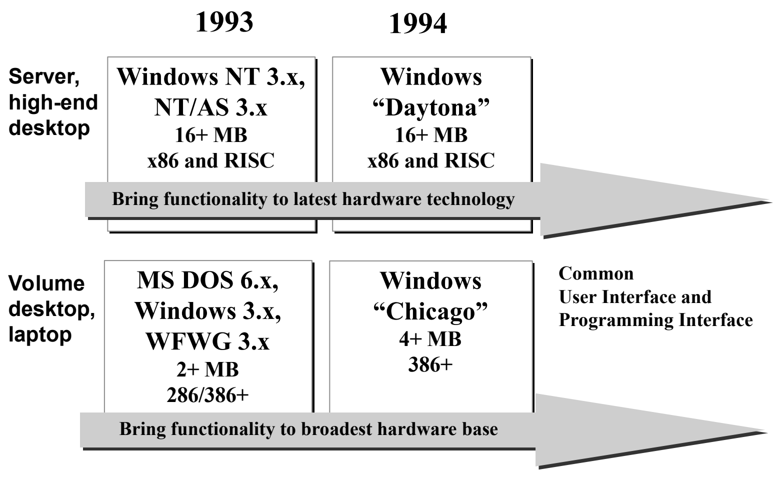

1993
1994
Server, high-end desktop
Windows NT 3.x, NT/AS 3.x
16+ MB
x86 and RISC
Windows "Daytona"
16+ MB
x86 and RISC
Bring functionality to latest hardware technology
Volume desktop, laptop
MS DOS 6.x, Windows 3.x, WFWG 3.x
2+ MB
286/386+
Windows "Chicago"
4+ MB
386+
Common User Interface and Programming Interface
Bring functionality to broadest hardware base

Figure 2. Evolution of Windows Operating System

Why Is There A Choice For a Client Operating System?

Microsoft offers more than one option for client operating systems because there are two distinct client markets—one for those using the mainstream systems and another for those using on high-end computers. No single operating system can fully optimize every hardware combination possible on with today's broad list of hardware choices. For the mainstream system (currently represented by products such as laptop and desktop computers), the primary operating system design goal is to deliver responsive performance for a broad range of applications while conserving the amount of system resources used. On the high-end (such as a RISC technical workstation or multi-processor server), the operating system must fully capitalize on the capabilities of the hardware and provide advanced services for the most demanding business and technical workstation applications.

Over time, as mainstream computers become more powerful, technologies implemented first on the high-end Windows operating system product will migrate to the mainstream product. Sometimes technical innovations will appear first on the mainstream product, due to timing of releases or because some features are focused on ease of use for general end-users (as is the case with Plug and Play support). The guiding principle for product planning is for the high-end product to provide a superset of the functionality in the mainstream product. Any deviations from this principle are temporary, due to variations in product release schedules

Which Client Operating System Should You Choose?

The answer to this question depends on what capabilities you requires. The Chicago (the next version of Windows) and Daytona (the next version of Windows NT) client operating systems each provide the following:

- Powerful operating system capabilities, with 32-bit preemptive multitasking, multi-threading and support for the Win32 API and 32-bit OLE 2.0
- Integrated connectivity support, with an open networking architecture for making simultaneous connections to multiple networks, built-in messaging capabilities, and remote access support
- Systems management capabilities, with the Windows Registry, user-level security, user profiles, backup agents, performance monitoring and event logging tools, and remote configuration support

Chicago provides all these capabilities while delivering the highest level of compatibility possible for applications and devices currently used with Windows 3.*x* and MS-DOS. However, Chicago also adds improved ease of use, with Plug and Play support and a new, more intuitive user interface.

Daytona enhances the base client operating system capabilities by providing:

- The highest level of capacity for advanced applications which have previously only been available on expensive UNIX and minicomputer platforms
- Support for multiprocessor and RISC systems
- The greatest degree of protection for applications and data, with access privilege restrictions for individual workstations, Uninterrupted Power Supply (UPS) support, a transacted file system, and support for multiple virtual DOS machines (MVDM) to allow preemptive multitasking of Win16 applications in separate address spaces

Chicago Evaluation Criteria

With the first technical Beta-1 release of Chicago, users and members of the press can begin reviewing or evaluating the features and functionality that Chicago brings to the industry. To effectively evaluate Chicago, it is necessary to not only know the feature set of the product, but also to understand the goals of the development team and the needs of the marketplace that have evolved into the development of the next generation of the Windows operating system.

To review or evaluate Chicago, you should approach the process in the following way:

1) Read this section of the *Chicago Reviewer's Guide* to understand Chicago from the point of view of providing a leadership operating system for the desktop and mobile computer user.

2) Read the rest of the *Chicago Reviewer's Guide* to understand the goals behind the Chicago project, as well as to understand the feature set and improvements that Chicago represents over Windows 3.1. Chicago is a large project, offering many features and benefits to end-users as well as corporate MIS organizations—this guide discusses these features and benefits.

3) Read the *Beta-1 Release Notes* for the latest information about the Beta-1 release. Chicago Beta-1 represents work in progress and the release notes will identify known bugs or incomplete feature sets to help testers avoid known problems that may impair your use of the Beta-1 release.

4) Since performance and compatibility issues are continually refined throughout the development process, wait until the final shipped product to properly evaluate Chicago's performance and compatibility. See "Areas Still Under Construction" for more information.

5) ***Try it!*** Once you've had an opportunity to explore how Chicago makes the PC even easier to use, you'll see why the Chicago development team is so excited about this product.

Areas Still Under Construction

The Beta-1 release of Chicago represents the first technical Beta release. While most of the base functionality is present in the Beta-1 release of Chicago, there are still several areas that are under construction and their present state is not entirely representative of the final Chicago product. For all practical purposes, Chicago as a whole is under construction and various components are likely to change or be enhanced.

It is through feedback that we receive from Beta sites that we drive the development efforts, bug fixes, and functionality enhancements between now and the time that Chicago is shipped as final product. It is also based on feedback from Beta sites and analysis of the bugs that are reported that final determinations are made as to when Chicago will be released to manufacturing.

One of the purposes of the technical Beta release of Chicago is to put the product in the hands of experienced MS-DOS and Windows users and have them provide feedback to Microsoft about compatibility with their existing applications, device drivers, and environment. While compatibility testing is being performed within the test labs at Microsoft, it is through a technical Beta release that the product can be tested with a breadth of different application programs, network operating systems, MS-DOS–based device drivers, Windows–based device drivers, and hardware devices.

The following areas provided as part of the Beta-1 release will continue to receive close scrutiny by the Chicago development team. We ask that you keep this in mind while examining Chicago:

- **Performance**

  Use the final released version of Chicago code, rather than the Beta-1 release, to formally test and report performance. Discussing relative performance improvements or degradations of a pre-final version of Chicago can be misleading to end-users and may not properly reflect the true performance levels they will observe when they try Chicago for themselves with the final product. Performance optimization is something you can look forward to in the final product. In a Microsoft product development cycle, implementation of features and functionality are completed in the early stages of development; code optimizations and performance tuning are refined towards the latter phases.

- **Compatibility**

  Use the final released version of Chicago code, rather than the Beta-1 release, to formally test and report compatibility. Again, this is an area that is best tuned in the latter stages of development and will not be accurately represented for evaluation in the Beta-1 release. Through the Beta test process, feedback from users about running their set of applications under Chicago is used by the development team to ensure that Chicago meets its final compatibility goals. Any issues related to compatibility that are identified while testing Chicago that results in behavior different from that of MS-DOS or Windows 3.1 should be reported as a bug as part of the Chicago Beta program.

- **User Interface Transition for Windows 3.1 Users**

  The initial efforts for developing a new easy-to-use interface for Chicago is focused at new and novice users. During the time after the Beta-1 release of Chicago and the next wide-scale Beta release, efforts will be undertaken by the shell interface and development teams to implement migration aids to make it easier for Windows 3.1 users to get up to speed with Chicago. Once advanced and power users overcome the initial learning curve of the new Chicago user interface, they will become quickly familiar and comfortable with the additional functionality offered by the Chicago shell. The usability lab at Microsoft and the interface designers continue to work towards decreasing the time it takes to transition experienced Windows 3.1 users, allowing them to discover the power, flexibility, and robustness of Chicago.

Suggested Evaluation Criteria for Desktop Operating Systems

As you compare Chicago with other operating system products on the market, including Windows 3.1, there are several areas that you should examine. Consider these areas to help identify the operating system that best meets your needs and the needs of users:

- Ease of use
- Performance
- Compatibility of device and application support
- Support for new applications
- Support for networking and connectivity
- Support for manageability and administration
- Support for communications and messaging
- Support for mobile services and remote access

The following sections discuss each of these areas in relationship to how Chicago provides the best desktop operating system for mainstream platforms.

Ease of Use

When looking at the ease-of-use aspects of an operating system, it is important to look at it from the perspective of both a novice and experienced user. Novices include those who have never used a PC, and those who use it infrequently, often because they find PCs intimidating. Typical problems for novices include finding it hard to navigate through the user interface and needing more information or coaching (for example, from online Help). Experienced users interact with more areas of the operating system than a novice user. These users demand flexibility, speed, and power.

As you evaluate how easy the operating system is to use, it's helpful to answer these questions:

- Is the operating system easy for a novice user to complete common tasks such as starting new applications, switching between two or more active application, or manipulating files?

- Is it easy for novice users to find the help they need to complete common tasks?
- Is the operating system flexible for experienced users to customize and tailor to the way they interact with the computer?
- Does the operating system allow the experienced user to be able to examine the configuration of their system, tweak or optimize their system, or observe the relative performance of their system?

Performance

System performance refers to how the operating system performs overall while running a set of broad tasks (for example, a group of applications and programs you would normally run simultaneously). Performance also refers to the ability of individual system components or subsystems to perform a more narrow set of operations (for example, file I/O).

Several suites of benchmarks are available that test the ability of operating systems to complete a set of tasks that are designed to mimic real world use of the PC and operating system combination. These benchmark tests produce numbers that represent the responsiveness of the operating system for a given set of commercially-available applications. It's possible to run the same suite of applications in your environment and use this information as a baseline to identify the relative performance between the operating systems you compare. However, benchmark suites don't tell the whole story and should be used in conjunction with traditional component-level benchmark tests and actual application tests.

In addition to running application suite benchmarks, it's best to isolate and separately test various components and subsystems of the operating system. This way you can obtain low-level results indicating how well the operating system can support services that are used by applications. Areas of performance that are commonly isolated and benchmarked on standalone PCs include the performance of the local file system for disk and file I/O, the performance of the graphics subsystem and video display drivers for graphics and text I/O, and the performance of the printing subsystem for printing I/O. In addition, desktop operating systems should be tested in networked environments for their ability to support network I/O throughput for the supported network clients, as well as the responsiveness for server functionality if supported by the operating system.

Of course, any operating system will perform best when your PC has the maximum amount of RAM. This configuration, however, is unrealistic since most end-users' PCs have less than the maximum amount of RAM. Performance test suites should be run against different hardware platforms including memory ranges from 4MB to 16MB, on platforms that include PCs containing Intel 80386DX, 80486, and even Intel Pentium-based CPUs.

When you evaluate performance, it's helpful to answer these questions:

- How well does the operating system complete a benchmark test exercising a suite of applications on a given hardware platform?
- How well does the operating system complete benchmark tests exercising individual components and device drivers provided as part of the system?
- How well does the operating system perform for network connectivity for supported network clients or provided network server functionality?

Compatibility: Device and Applications Support

When it's time to replace an old operating system, a key question to consider is "Can my company still use our existing applications and hardware with the new operating system?" If you are thinking about replacing an existing operating system, your company probably has invested a large amount of money in applications, printers, modems, and other PC-related peripheral devices. Now it's important to find out if the replacement operating system can run with your existing hardware and software.

It's also important to consider how broad a range of devices is supported by the operating system you choose. No doubt, as your company grows, your hardware needs will grow too. Your choice of an operating system should not unreasonably restrict the peripheral devices your company can buy later. The operating system you choose should include ample device drivers, not only to support the devices you currently own, but for those you will buy in the future.

When examining device support of an operating system, consider the number of devices supported, the industry standards that the operating system supports, and the compatibility for using existing device drivers shipped with earlier operating systems or with devices themselves.

As you evaluate the device and application support in an operating system, it's helpful to answer these questions:

- Does the operating system provide broad device support for existing hardware and associated MS-DOS and Windows–based device drivers?
- Are devices easily recognized, installed, and configured by the operating system?
- Does the operating system support running your existing MS-DOS or Windows–based applications as well as MS-DOS 6. *x* or Windows 3.1?
- Does the operating system allow easy exchange information between applications or support advanced inter-application communication mechanisms?

- Does the operating system provide services for new types of applications (for example, multimedia, remote access, and communications-related applications) ?

Support for Network Services and Connectivity

In a corporate environment, it's extremely important for an operating system to be able to provide network support for a broad base of clients. An operating system that can support connectivity in a heterogeneous environment distinguishes that operating system from the others. Likewise, it's important to compare how well network functionality and other areas of the system (such as the user interface) are integrated in each operating system.

In general, it's more than just incorporating proprietary network functionality in an operating system product that customers are looking for. Customers also want support for industry-wide standards to prevent reliance on a single vendor to support a multi-vendor environment.

As you evaluate the networking support for an operating system, it's helpful to answer these questions:

- Does the operating system have built-in, native support for popular networks?
- Does the operating system natively support a wide range of network transports (such as TCP/IP and IPX/SPX), industry-wide communication protocols (such as RPC, NetBIOS, DCE, named pipes), and existing network device standards (such as NDIS and ODI)?
- Does the operating system provide a simple, consistent user interface for accessing the network and using network resources?
- Does the operating system support an open architecture to allow third-parties and network operating system vendors to easily integrate or add network connectivity enhancements or application support?

Support for Manageability and Administration

PCs are now one of the largest expenses of an MIS organization. Medium and large businesses invest tens of thousands of dollars each year, not only on the hardware and software for new and existing computer systems, but also for setup and administration of these systems. Thus far, there is little consistency and almost no integration among available tools for managing and administering PCs in a networked environment.

The good news is that standards organizations are now working to simplify the management scenario by developing standard ways for managing PCs. These standards will mean better and more integrated management tools for the network administrator. For an administrator to reap any benefits, however, it's important to choose an operating system that supports management mechanisms adhering to existing standards or one whose infrastructure is designed for adaptability to a new standard.

As you examine the support for manageability or administration of a desktop operating system, it's helpful to answer the following questions:

- Does the operating system provide a platform for supporting existing industry standard management mechanisms such as SNMP, and provide the flexibility for supporting future standards such as DMI?
- Does the operating system provide tools and mechanisms for MIS organizations and administrators to customize and control the functionality and capabilities on the desktop?
- Does the operating system provide support for administering or managing the desktop PC remotely over a network?

Support for Communications and Messaging

The explosive growth of services such as CompuServe, America Online, and the Internet show the dramatic increase in demand for operating system that enable connectivity beyond the desktop to access online and mail services. The support and services that an operating system provides can open the door for users to discover the communications and messaging possibilities in the Information Age.

As you evaluate communications and messaging support in an operating system, it's helpful to answer these questions:

- Does the operating system support high-speed communications and background multitasking capabilities?
- Does the operating system provide support for my communication hardware, and provide services for supporting new communication functionality such as sharing communication ports, unified device configuration, or support emerging communications technology?
- Does the operating system provide support for industry-standard messaging services?
- Does the operating system provide broad communication and messaging capabilities and consolidated information access (for example, faxing, dial-in access to resources, and access to online information services) ?

Support for Mobile Services and Remote Access

To realize seamless mobility, users must be able to easily communicate and remain productive, whether they are in the office, at a customer site, or at home. Users must be able to communicate with coworkers and clients regardless of their location. Additionally, transitions home computer to portable computer to office computer must not cause interruption in a user's workflow. Support for mobility services as part of the operating system ensures tight integration and connectivity between portable computer and desktop PCs, allowing minimal interruption from real work as a user switches from one location and/or computer to another.

As you evaluate the support of an operating system for mobile services, it's helpful to answer these questions:

- Does the operating system support remote access to the key services or information you need on your corporate network?
- Does the operating system have robust support for the dynamic nature of mobile hardware, such as PCMCIA, power management, and docking stations?

Chicago Product Overview

Chicago is an extremely feature-rich product. Virtually every aspect of Chicago reflects improvements over Windows 3.1. This guide is organized by technology area present in Chicago, and provides an overview of the features, functionality, and components that make up Chicago.

While reading about Chicago, it is important to put the functionality offered into perspective of the needs of the marketplace with how Chicago its target design goals.

Chicago Product Areas Covered by Guide

This guide presents an overview of Chicago by discussing the areas of technology that make up Chicago. To facilitate a discussion of the product, the following areas of Chicago that are examined in this guide include:

- The Chicago User Interface
- Base System Architecture
- Robustness Improvements
- Improved Support for Running MS-DOS–based Applications
- Plug and Play
- Improved Device Support
- Networking
- Desktop Systems Management
- Printing Improvements
- Communications
- Mobile Computing Services
- Chicago’s Messaging and Information Center
- Multimedia Services
- Installation and Setup of Chicago
- International Language Support
- Accessibility
- What Makes a Great Chicago Application?
- Chicago Questions and Answers

Summary of Improvements over Windows 3.1

A summary of improvements available in Chicago over Windows 3.1 is provided in many of the sections of this guide covering key features and functionality available in Chicago. This section provides a quick overview of area where Chicago address Windows 3.1 issues, or improves upon the based functionality.

Try It!

Where appropriate, each section of this guide includes a Try It! Section that gives you the opportunity to try and examine the areas of improvement in Chicago. You are encouraged to see for yourself that Chicago is a more flexible, powerful, and robust operating system than Windows 3.1. ***Try It!***

Under Construction...

Not all of the functionality that will be available in the final Chicago product is provided in the Beta-1 release. While major changes in base functionality in Chicago are not expected between now and the next release, Microsoft is continuing to conduct research, elicit feedback from beta testers, and refine functionality and features. Elements of the system including the UI, device support, and packaging details are unfinished and will change or be refined before the final shipment of Windows “Chicago.”

While this guide provides a good overview of the Chicago product, components such as applications, utilities, and some other support functions are not discussed as they are not present in the Beta-1 release and are still being worked on.

The Chicago User Interface

When you first boot Chicago it is immediately apparent that the old world of Windows running on top of MS-DOS is no more. Gone are the character-mode boot messages that held meaning only for a very small minority of computer users. Instead, you are graphically carried to the desktop of the new Chicago user interface (UI).

More than any other part of the operating system, the UI defines a user's overall experience. The easier, more powerful, and more compelling the UI, the better the user feels about computing which, in turn, makes the user more productive. Great UI helps grow the industry by making computing easier and more natural for *all* people, from the new user to the power user. This is the mission of the Chicago UI.

This section introduces you to the Chicago UI and its conception. It is divided into the following topics:

- **Objectives**. Lays out the top-line goals of the Chicago UI.
- **Methodology**. Overviews the design-test-redesign loop that has been critical to the UI development process.
- ***Easy***. Outlines features that make Chicago easy to learn and use, especially for those new to Windows.
- ***Powerful***. Outlines features that make Chicago more powerful, efficient, and customizable for the experienced Windows user.
- ***Compatible***. Outlines features that make Chicago easy to learn and use for those familiar with Windows 3.1. Changes to the UI between Beta-1 and general release will focus on this important area.

Designing the Chicago User Interface

Objectives

The overarching goal of the Chicago UI is to make PCs even easier to use for *all* people.

Fulfilling this goal is a challenge because people work in very different ways. For the beginner, performing a task must be easy to learn even at the expense of efficiency. However, the experienced user is interested in doing more with the PC and in efficiency and flexibility. In addition, the user who upgrades from Windows 3.1 must not be made to throw out everything he or she has learned.

Chicago has fulfilled these disparate needs by making the most common and essential features of Chicago (such as, launching an application, task switching, finding a file) easily discoverable by the beginner via the Chicago Taskbar, with its

"Start" Button and push button task switching. At the same time, the product is deep in power user capabilities that promote efficiency, customizability, and control such as the "Explorer", rich secondary mouse-button clicking capabilities, properties sheets, and "Shortcuts".

The Chicago UI is designed to be scaleable—that is, to fit the proficiency and preferences of the individual user.

Methodology

The Chicago UI was not a grand plan designed to a master specification. It started out with clear objectives, guiding design principles, and a skilled team. The design process has been full of discarded designs, new ideas, and a great deal of learning.

The process started with answering the basic question: *How can Windows 3.1 be improved*? From there the UI team began to work through a design, to usability test, to redesign loop that continues to iterate today.

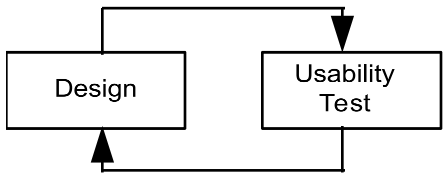


Figure 3. Chicago feedback design loop

Improving Windows 3.1

There was no shortage of information sources in determining how the Windows 3.1 UI might be improved. The following table summarizes key findings.

How can the Windows 3.1 UI be improved?

Make it easier to learn for novices. Problem areas:

- Window management (overlapping and minimized windows) is confusing
- Hierarchical views (like the file manager) are confusing
- Double-clicking to launch applications is not discoverable
- Task switching is undiscoverable, which means too few users are taking advantage of running multiple applications

- Make it more efficient and customizable for experienced users. Problem areas:
- Too much "middle management". Confusing and overlapping functionality: Program Manager, File Manager, Print Manager, Windows Setup, Control Panel
- 8.3 file naming
- Not customizable
- Poor network and connectivity integration

Figure 4. Goals for improving the UI over Windows 3.1

We used the following mechanisms to compile feedback data:

- **Usability Tests**. The Microsoft Usability Lab, detailed below, is primarily used for testing usability of new designs. However, in order to better understand how people are using Windows 3.1 today and to establish a baseline, several phases of testing were dedicated to Windows 3.1.

- **Focus Groups**. Several focus groups were conducted with different levels of user to determine the problems people are having with Windows 3.1 today.

- **Educator Feedback Program**. Last year a team of UI designers and testers visited 12 independent software education companies. More than any other people, software educators understand the everyday usage challenges faced by beginner and intermediate users. Questions like *"What are the 5 hardest tasks for students to learn in Windows?"* and "*What 5 changes would you make to Windows to make it easier to learn?"* were asked. These educators have also served as a great resource for testing Chicago prototypes.

- **Suggestion Database**. Thousands of UI suggestions from Windows 3.1 end-users and corporate customers have been compiled and analyzed. Going forward beta tester UI feedback will be incorporated into the final release UI.

Many of the advances you will observe in the Chicago Beta-1 UI are targeted at specifically solving the problems users are currently experiencing with Windows 3.1.

Putting New Designs to the Test

Conducting extensive live tests in a variety of settings with a variety of subjects has been key to the engineering of a state-of-the-art UI. A large portion of the total Chicago development budget has been expended on this critical activity. Chicago is likely the most usability-tested product ever. Methods employed for testing the Chicago UI follow.

- **Formative Testing in the Usability Lab.** Conducted primarily in the groundbreaking Microsoft Usability Lab, formative testing tests real users on specific tasks (such as, launching a program, finding a file, and installing a printer). The Usability Lab has nine testing suites, each with a one-way mirror, cameras, and other equipment for observing and recording users as they work. Central to the Lab operations is online data collection software which helps specialists collect cognitive and quantitative process data as subjects work through sets of tasks.

  Usability tests are observed first hand by the design team and are essential in future designs. To date, we have conducted *more than 1,000 hours of usability testing over 30 phases of lab testing with more than 250 participants*.

- **Summative Testing**. Conducted at customer sites and in the Usability Lab, summative testing involves testing of the UI as a whole with real users over longer periods of time. Several phases of summative tests have been conducted to date at corporate sites. As the product approaches shipping, usability tests will focus on summative testing.

- **UI and Industry Expert Review**. Last fall, a panel of UI and industry experts was assembled to review and critique the Chicago UI. Also, four individual consultants spent 1.5 days each “alone with Chicago” and gave extensive feedback.

Easy

"Easy" is a broad term so it requires some definition. This section will detail those features of the Chicago UI that are focused on solving the first category of "How can the Windows 3.1 UI be improved" from above. Namely, learnability for those who are new to Windows.

The Desktop: Neat, Clean, and Logical

After you boot into Chicago, you are presented with the new Chicago desktop (see Figure below). It's neat and clean with only a few graphical objects on the screen. It's like moving into a new office before you have the chance to really get it messy.

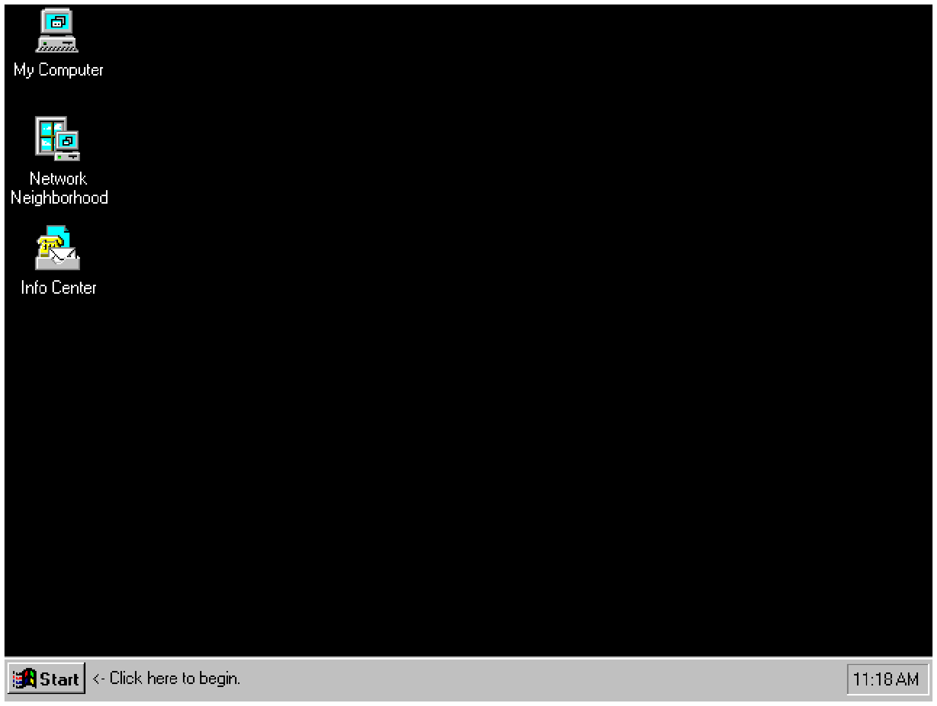


Figure 5. The Chicago Desktop

The simplicity of the desktop appeals to all users' sense of cleanliness but also serves to focus the novice user on the essentials:

- **Taskbar**. Quickly start a program from the Start Button. Easily switch tasks.
- **My Computer**: Makes browsing your PC logical and easy.
- **Network Neighborhood**. In the world of mapped drives and complex interfaces, users are unable to browse the network. Chicago's Network Neighborhood makes browsing networks possible and easy, independent of

the network provider (such as, Windows NT Advanced Server, NetWare, or Chicago itself).

- **Info Center**. Optionally installed. Gives the user a single place to go to access all MAPI-provided information (such as, Mail, Microsoft At Work™ faxing).

The Chicago Taskbar: Home Base

More than any other feature, the Taskbar exemplifies the order of magnitude improvement in ease of use and learnability of the Chicago UI. It is the anchor of the UI. Its mission is to make 95% of what a typical user wants to do with the operating system easily accessible at all times. An indicator of a great design is that it turns out to be much more than it was originally intended. The Taskbar started out specifically as a novice user program launcher and task switcher. However, its simplicity and power have turned out to be favorites of experienced windows users, and it has many more capabilities.

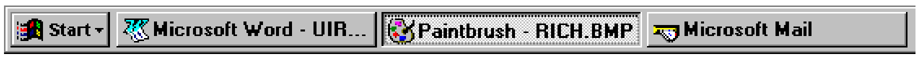


Figure 6. The Chicago Taskbar

The two key features of the Taskbar are the Start Button and Push-button task switching.

The Start Button: Up and Running in Seconds

Usability tests on Windows 3.1 show that it takes a brand new Windows user an average of nine minutes to open “Write”. With Chicago, opening Wordpad takes a new user an average of three minutes. If only the users that launched Wordpad via the Start Button (rather than by other means) are counted the average time to launch drops below one minute! The main reason for this dramatic 3x-9x speed improvement is the Start Button. Without ever having to know about double clicking, complex hierarchies, or program manger groups, a beginning Chicago user can quickly launch a program and get to work.

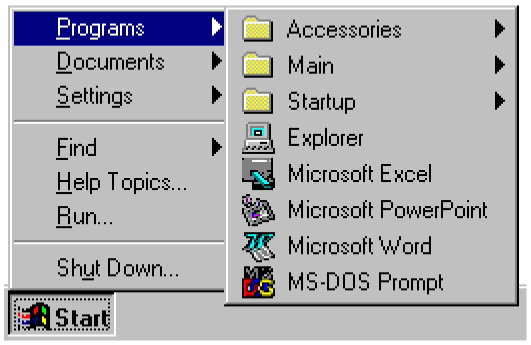


Figure 7. The Chicago Start Button

However, the Start button is much more than a super-efficient program launcher.

- **Programs**. During Setup the user is asked to select his or her most often used programs. These programs are placed in the Programs menu of the Start Button. In the future the user can easily change the programs that appear on this menu by selecting Taskbar Settings right from the Start Button. For upgrades, all of their Windows 3.1 program groups are converted to folders within the programs folder and are accessible from the Start Button.
- **Documents**. The Documents menu of the Start button contains a list of the last 15 documents the user opened. It provides very quick access to the most recently edited files. This helps prevent time-consuming and frustrating browsing and helps people begin to think of their work in terms of documents ("document-centricity"), rather than applications.
- **Settings**. Gives quick access to the Control Panel, the Printers folder, and the Fonts Folder. It also allows the user to customize the Taskbar itself (such as, what programs to include in Start Programs menu) to suit personal working preferences.
- **Find**. Find is a new feature of Chicago that goes far beyond File Manager's File Search feature in Windows 3.1. Searches do need not conform to the *.* searching syntax, and criteria such as last modification date, size of file, and full text can now be used. More on Find in "Power" below.
- **Help Topics**. Help has been overhauled for usability in Chicago and is easily accessible from the Start menu. See "Help" topic later in this section for details.
- **Run**. Provides enhanced command-line type functionality from the Start Button.
- **Shutdown**. Allows for easily accessible and safe shutdown, restart, and logoff.

Try This—Customizing the Start Button

Mouse

- Select Start, Settings, Taskbar.
- From the Change Start Menu property sheet, choose which programs you'd like to appear either at the first level of the Start Button or inside the programs group.
- Close and check your new configuration by pushing the Start Button.
- **Power user hint:** You can also put a program on the Start button by dragging and dropping a Shortcut to the program right on the button.
-

Try This—Test a Novice

Mouse

- Get a stopwatch.

- Find a friend or family member who is a computer novice. Sit this person down at your Chicago PC with nothing running and a clean desktop.
- Ask him or her to open an application called “X”, where X is in the programs group. Note the time to completion.
- Try the same thing on Windows 3.1.
- Compare.
-

Mouse

Try This—CTRL+ESC Brings Up the Start Button

- Select CTRL+ESC and the Start Menu will pop up.
-

Task Switching Made Simple From the Taskbar

Novices need powerful features presented to them in a very simple and compelling way, otherwise these features will not be used. Research on active Windows users shows that only 27% of general Windows users frequently use more than one application at a time and only 20% frequently use ALT+TAB task switching. These powerful features of Windows 3.1 are simply not discoverable.

The objective of the Taskbar is to make switching among multiple applications as simple as changing channels on a television set. Every new window that is opened automatically gets a button on the Taskbar. To change tasks, all the user must do is go to the Taskbar and select the desired channel. No more minimized program icons, no more disappearing windows. No matter where the user is, he or she can see all of his or her active tasks simply by looking at the Taskbar, the Windows TV guide.

Task Buttons re-size automatically depending on the number of active tasks. Should the buttons get too small to be useful the user can custom configure the Taskbar. In fact, there are a host of Windows Taskbar configuration options that allow the user to configure it to fit his or her needs including:

- **Reposition**. The Windows Taskbar can be dragged to any perimeter position on the screen.

- **Re-size.** The width of the Windows Taskbar can be widened by dragging the inside edge.

- **Auto Hide**. The Windows Taskbar can be hidden from the screen and made appear only when the mouse hits the screen edge by selecting Settings, Taskbar from the Start Button.

Also, noteworthy is the animation when a task is minimized into the Taskbar or maximized from the Taskbar. It helps new users understand “where” a program goes when it is minimized.

An Easier Model for File Management and Browsing

File management and browsing in Windows 3.1 was not intuitive. Fewer than 55% of general Windows users regularly use the File Manager. For novice users the File Manger is especially confusing and intimidating.

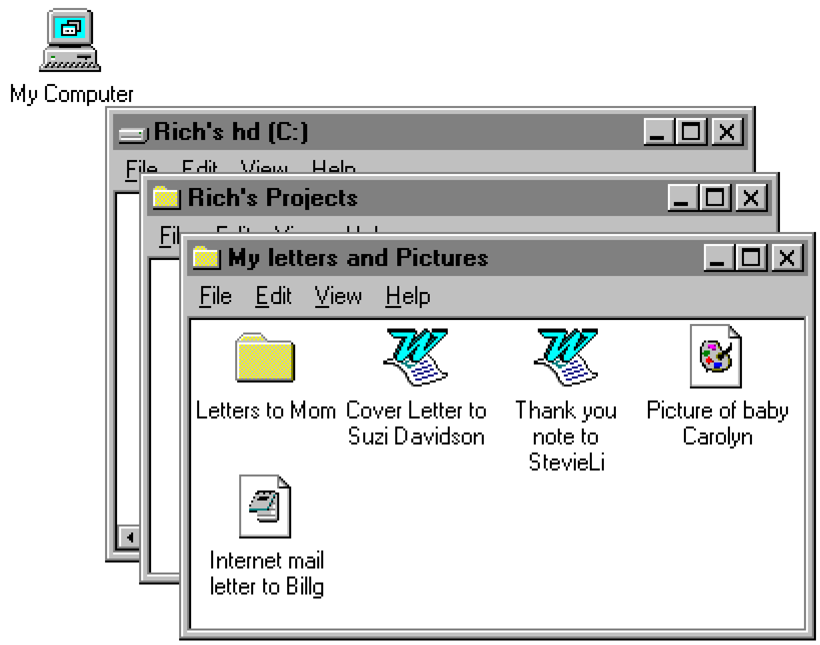


Figure 8. Browsing My Computer

New Windows and Large Icons Work for those new to Windows

Designing a discoverable and comfortable model for browsing and file management for the novice user has been a priority for the UI design team because of the observed difficulties with Windows 3.1. Several significantly different designs have been tested and thrown out. In the course of this testing the design team made a few basic discoveries about file management and browsing:

- Exposed hierarchies are intimidating and unintuitive.
- Dual-pane views (hierarchy on the left, contents on the right) are also intimidating and unintuitive. Novices have difficulty understanding the connection between the logical tree hierarchy on the left and the contents pane on the right.
- Object-Oriented UI is great for basic tasks, but not for complex ones. There exists a general belief that the more object oriented a UI is the easier it is

for the user. This is an appealing theory, but in real life this is not the case. Direct manipulation of screen objects and logical resulting behaviors are important for basic functionality (such as, dragging a file from a folder to the desktop). However, advanced direct manipulation features such as dragging a file to a printer icon, are not intuitive. Intuitively, users understand selecting an object with the mouse then browsing menus or buttons for actions to perform on that object.

- Large icon views are much more comfortable than list views.
- A novice's ability to find what he is looking for and feeling comfortable and "grounded" along the way are the defining characteristics of a good browsing experience. Efficiency and speed are less important.

The "My Computer" default browsing model is the result of all of this design, testing, and learning. A folder or drive can be opened by double clicking or selecting it and choosing File Open. The default browsing model brings up a new window in large icon view. To many advanced users this behavior seems cumbersome. *Why not open in list view? Why create a new window, it just clutters up my screen? Why not open to a dual pane view? It's much more efficient for me. Why not turn the Toolbar on by default?* All of these models and more were tested thoroughly and discarded (as the default configuration) because they caused confusion and stress among novices. Novices respond best when presented only with essential information and when they can easily "get back" to where they just were.

Note Multiple configuration options are available to experienced users in View Options.

Chicago has a very powerful dual-pane browsing application for Experienced users called the Explorer, which is likely how you, as an experienced user, will prefer to browse. The Explorer will be covered in "Power" below. Additionally, the File Manager can be run for backwards compatibility.

New Capabilities in the Default Browsing Model

New capabilities of the default browsing model should not be overlooked in this discussion of simplicity. Folders can be created within folders. Files and folders respond very logically to being dragged and dropped. Files and folders can be cut, copied, and pasted just like text and objects within applications. Views can be customized by the user and each window "remembers" how the user last configured it, so that the next time it opens it is in the user's favorite view. The best way to discover the capabilities of the default browsing model is to play with it yourself, or better yet, find a novice user and watch him use it.

Try This—Turn Off "Open into a new window"

Mouse

- Double click My Computer.
- Select View Options.

- From the Folder property sheet, choose Browse Folders With a Single Window.
- Turn on Toolbar by selecting View Toolbar.
- Now double-click your hard drive—no new window.
-

Name Files in English with Long Filenames

By far, the number one most requested feature since Microsoft has been in the operating system business is long filenames. The usability win by eliminating the need to conform to the 8.3 naming convention is obvious and large. To ensure backwards compatibility with the universe of existing MS-DOS and Win16 applications, extensions have not been eliminated, just hidden from view by default.

Figure 9. Chicago long filename

Additionally, files can be renamed in place in Chicago by selecting the file, clicking on the filename, and typing a new name. The hidden file extension is not affected by renaming the file. Files can also be renamed from within the new Chicago common dialogs (including File Open and Save).

Try This—Show File Extensions

- Users who want to view extensions can do so from any folder.
- Choose View Options.
- Select the View tab.
- De-selecting Don’t Display File Extensions.
-

Network Neighborhood and Networking Accessibility

This section will discuss how the Chicago client makes browsing networks possible and easy, independent of the network provider (such as, Windows NT Advanced Server, Netware, or Chicago itself). For more details about Chicago’s networking capabilities, see the section called “Chicago Networking and Systems Management.”

The Network Neighborhood icon, shown in the figure below, sits on the desktop and logically separates for the user the place to go to browse resources not on “My Computer”. The user can easily browse the network via the Network Neighborhood just as if he or she were browsing his or her hard disk.

Figure 10. Network Neighborhood desktop icon in Chicago

- **The Network Neighborhood** is also configured by the administrator to display, at the top level only those PCs, servers, and printers that are in the user's immediate workgroup. This insulates the user from the vastness of large corporate networks. However, if the user wants to browse the larger network, this can be done by opening "Entire Network" from within the Network Neighborhood. This was not possible prior to Chicago. When a user browses servers, network connections are being made without ever having "mapped" a drive.

Try This—Shortcut to a Network Folder on the Desktop

- Browse the network neighborhood until you find an often-used network folder.
- Hold down right mouse button and drag and drop that folder on the desktop.
- Choose Create Shortcut Here.
- Close the network window.
- Double click the shortcut and your network folder opens into a new window. This will be available every time you boot into Chicago. For more information on shortcuts see the "Powerful" section below.
-

- **System-wide support for UNC pathnames** makes obsolete the unnecessary process of "mapping" drives (assigning new drive letters to a specific network resource). This technology allows the natural network browsing observed through the Network Neighborhood. UNC pathname support allows a whole host of usability improvements of which network browsing is just one.

Try This—UNC "Run" to Your Favorite Network Folder

- From the Start button choose Run.
- Type the full UNC path to your favorite network folder (such as, "\\MKTG\PROGRAMS\SARAHB") and press ENTER.
- The folder will open up in a new Window. No drive mapping.
-

- **The "Network" Control Panel tool** consolidates all networking configuration in one location. Solves difficulty of configuring Windows networking under Windows 3.1 and Windows for Workgroups 3.*x*.

- **Easy drive mapping** is also available in Chicago. There is a Map Network Drive button on the Explorer and browsing window toolbars. Also available via right-click on “My computer” for power users. Mapped drives appear as persistent connections in “My Computer”.

- **Networking and mobility** are intrinsic to the Chicago UI. The Chicago UI was designed from the ground up with networking and remote access in mind. For example, when a file copy detects that the copy is being performed over a slow-link (modem connection), the copy dialog itself includes an “estimated time to completion” clock.

- **Networking integration with new common dialogs** (including File Open and File Save). Full exploitation of new common dialogs throughout the system will not be implemented until Beta-2. However, key to the great leap in usability of new common dialogs observed in the labs, is tight integration with networking. From new common dialogs, the Network Neighborhood can be browsed just like My Computer. Also, the majority of basic file management tasks can be performed from within common dialogs.

Try This—Create a New Folder From Within Common Dialogs

Mouse

- Click the Start button, then choose Programs, Accessories, System Tools, Admin Configuration Tool. Admin Config is a new network and system administration tool that happens to takes advantage of new common dialogs in Beta-1.
- Select File Open.
- Choose the Create New Folder icon.
-

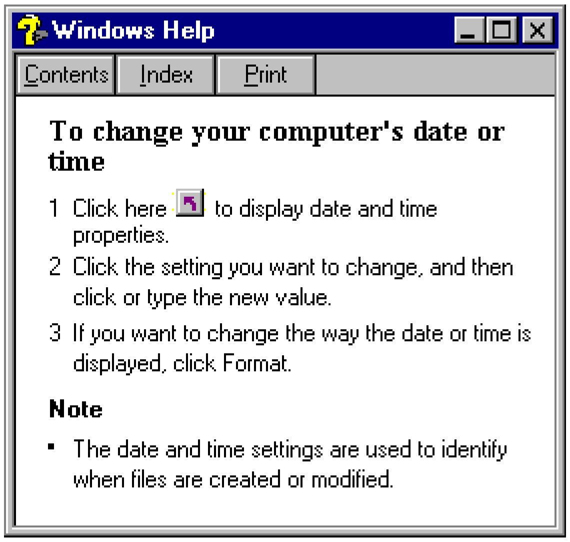
New Help Engine: Accessible and Useful Online Information

Online help has been completely re-tooled in Chicago. It underwent extensive usability testing in the labs and the result is a significantly easier to use and learn help system. Additionally, customizing and developing Windows help files by ISVs and corporate customers has been made dramatically easier. A brief description of the major features of new Chicago Help follows.

- **Simplified interface**. Help in Windows 3.1 was difficult to learn and use. It had three main functions: Contents, Search, and Glossary. The Contents view was not well organized and presented and there was some ambiguity about which of the functions to use when. Chicago behaves much more intuitively and more like a real reference book. It only has two Tabs: Contents and Index.

- **The “Contents” Tab is organized like a book’s table of contents**. Top level “chapters” (iconically represented by a book) are displayed and can be drilled down on for sub topics (iconically represented as a page). Many chapters also have “Tips and Tricks” subsections. These have proved popular in lab testing.

- **Help Topics are short**. They all fit in one small screen to keep users from having to scroll through large, complicated help topics.



Windows Help

Contents | Index | Print

To change your computer's date or time

1 Click here to display date and time properties.

2 Click the setting you want to change, and then click or type the new value.

3 If you want to change the way the date or time is displayed, click Format.

Note

- The date and time settings are used to identify when files are created or modified.

Figure 11. Help Shortcut button

- **Shortcut Buttons make using Help advice simple**. New Shortcut buttons are the most popular feature of Help. Some Help topics contain these shortcuts that take the user right to the area of Chicago that it is referencing. For example, a user who is searching for help on how to change the clock on the PC can "jump" right to the Clock Control Panel tool, right from within Help. (see figure above).

- **What's This?** From within all Chicago Control Panel tools, a new "?" icon appears on the upper-right of the Title Bar. By selecting this the user's cursor changes to a "?" and can be dropped on any target in the dialog box. This brings up a short description of whatever was selected. "What's this?" can also be accessed by right-clicking within Control Panel tools.

Try This—Use Help's Shortcut Buttons to Change Desktop Color

Mouse

- From the Start button choose Help Topics.
- Select the Index tab.
- Type "**display**". Double-click the option called Background Picture Or Pattern Changing.
- Click the shortcut jump button to Display Properties tool.
-

More “Document-centric”

OLE 2 introduced document-centricity with in-place editing of objects. The application window changes and the document stays the same. This makes the software begin to work the way people work, rather than vice-versa.

Figure 12. New Word document template

The Chicago UI picks up on the concept of document-centricity in several subtle, but powerful ways including:

- **A window is just an open view of an object**. When the user opens a folder from anywhere in the UI, a new window opens up. The title of the new window is the same as the name under the folder before the user opened it. This is logical. In the next generation of applications written for Chicago, ISVs will follow this same model. A Word for Chicago document called “My document” is double-clicked from the anywhere in the UI, and a new window (Word itself) is opened entitled “My Document—Microsoft Word”. This is partially implemented in Beta-1 with Wordpad and Paint.
- **“New” templates from within folders and in the Explorer**. From within any folder in Chicago or from the desktop, new files can be created in place by selecting File New and then choosing a file type. This is very convenient for managing files based on projects rather than the whim of an application.

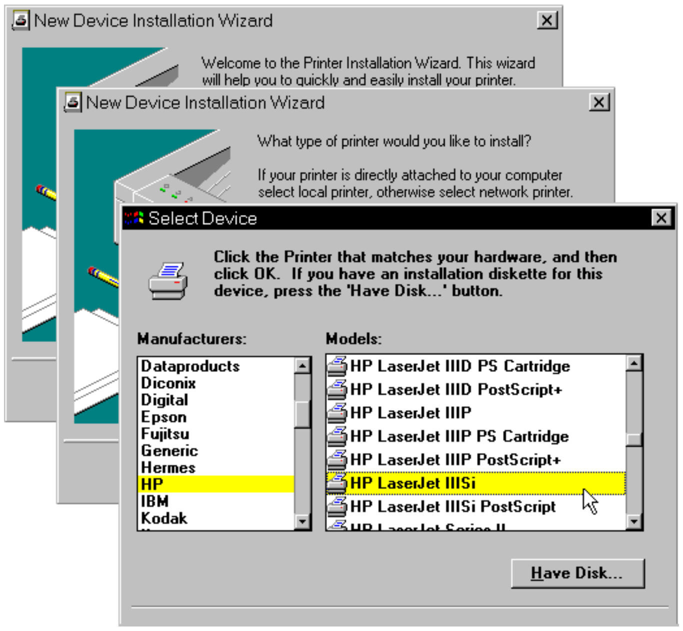
Try This—Create a new Wordpad document from within a folder

Mouse

- Browse to a project folder where you’d like to save a new Wordpad document.
- From the File menu choose New then Wordpad Document.
- Type a name and press ENTER.
- Double click your new document to launch it.
- **Hint:** This functionality can also be accessed by right-clicking from within any folder or the desktop.
-

Wizards: Your Guide to Powerful Capabilities

Started in Microsoft’s Applications Group, Wizards are a proven tool that make it easy for all classes of user to take advantage of powerful but complex functionality. A series of questions are posed to the user in a friendly and straight-forward way.


New Device Installation Wizard
New Device Installation Wizard
Select Device
Manufacturers:
Models:
Have Disk...

Welcome to the Printer Installation Wizard. This wizard will help you to quickly and easily install your printer.

What type of printer would you like to install?

If your printer is directly attached to your computer select local printer, otherwise select network printer.

Click the Printer that matches your hardware, and then click OK. If you have an installation diskette for this device, press the 'Have Disk...' button.

Manufacturers: Dataproducts, Diconix, Digital, Epson, Fujitsu, Generic, Hermes, HP, IBM, Kodak

Models: HP LaserJet IIID PS Cartridge, HP LaserJet IIID PostScript+, HP LaserJet IIIP, HP LaserJet IIIP PS Cartridge, HP LaserJet IIIP PostScript+, HP LaserJet IIISi, HP LaserJet IIISi PostScript

Figure 13. New Device Installation Wizard Walks User Through Installing a Printer

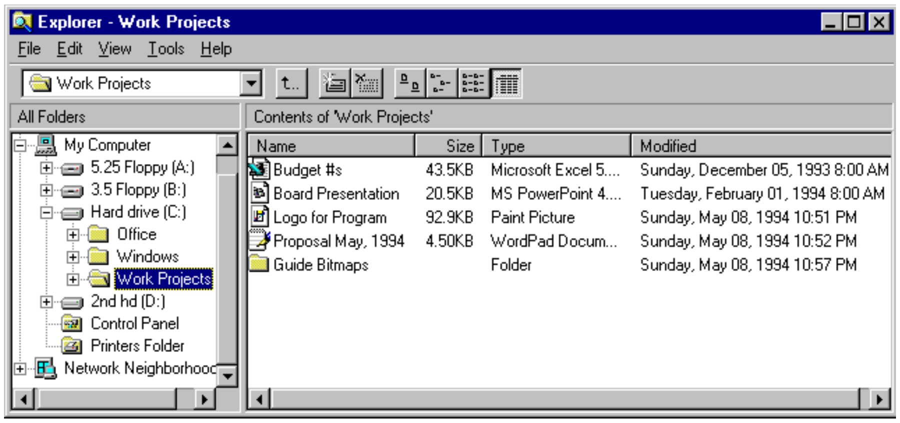
Chicago uses Wizards throughout the system, including:

- Add Printer wizard in the Printers Folder
- New Device wizard in the Control Panel
- Remote Access setup wizard in the Network Neighborhood

Powerful

Experienced users glean many of the same benefits from the Start Button and the Chicago Taskbar as do beginners—quickly launch a new program, quickly task switch, etc. However, experienced users need more. They need a powerful way to browse and manage file hierarchies be they local or somewhere else. They need to be able to customize the UI to suit their needs and tastes. They need to be able to take shortcuts to get tasks done more quickly and efficiently. They need to be able to do *more.* The new Chicago UI enables the experienced user to do more, as you will see in the coming pages and during you're own explorations.

The Explorer: Power Browsing and File Management



My Computer
- 5.25 Floppy (A:)
- 3.5 Floppy (B:)
- Hard drive (C:)
  - Office
  - Windows
  - Work Projects
- 2nd hd (D:)
- Control Panel
- Printers Folder

Network Neighborhood

Name	Size	Type	Modified
Budget #s	43.5KB	Microsoft Excel 5....	Sunday, December 05, 1993 8:00 AM
Board Presentation	20.5KB	MS PowerPoint 4....	Tuesday, February 01, 1994 8:00 AM
Logo for Program	92.9KB	Paint Picture	Sunday, May 08, 1994 10:51 PM
Proposal May, 1994	4.50KB	WordPad Docum...	Sunday, May 08, 1994 10:52 PM
Guide Bitmaps		Folder	Sunday, May 08, 1994 10:57 PM

Figure 14. The Chicago Explorer

One Chicago developer describes the Chicago Explorer as the "File Manger on steroids". It is powerful, flexible, efficient and extensible. It also solves many fundamental problems with the Windows 3.1 File Manager, like having to have a new window for every drive. For many Chicago power users the Explorer will be the primary interface. The best way to understand the Explorer is to experience it firsthand, however, here is an overview of its major features:

- **Single view on a world of information**. The Explorer is the eyes of the Chicago PC. With it, the user can view the whole of Chicago's single, unified namespace (all resources, local or connected) from 10,000 feet or zoom down to 10 inches. "My Computer" and the "Network Neighborhood" can be browsed and managed, and, if the MAPI 1.0 subsystem and Chicago Mail are installed, the "Infocenter" can be browsed giving access to mail, shared folders, MS At-Work faxing and any installed MAPI service providers (such as, CompuServe email).

- **Flexible and Customizable**. Via the Explorer toolbar and View menu, the user can view folder contents in several ways including large icon, small icon, list and details views. Folder contents can easily be sorted by name, size, type, and modification date by selecting the column title. The user can also map network drives from the Explorer toolbar.

- **Rich information about objects in Details View**. Details view provides a wealth of context-sensitive information about folder contents. For example:
  - Files retain their identifying icons
  - Drive sizes and free space (even mapped network drives) are reported in My Computer
  - Descriptions of Control Panel Tools
  - Jobs in queue in the Printers folder
  - Comments on others computers in the Network Neighborhood
- All of the powerful right-click and Properties features described in the following two topics are supported in the Explorer.

Try This—Copy a File to a Different Drive Without Opening a new Window

Mouse

- Right-click My Computer and select Explore. Maximize the window.
- Select a file that you would like to copy to a network or floppy drive.
- Mouse back to the left pane a the Explorer and use the "+" icons just to the left of folder and drive icons to find the network folder you wich to copy the file to. Do not click on the destination folder.
- Go back to the right pane where your original file is currently stored and drag & drop it to the destination folder.
- Operations like this could not be performed in the File Manager without opening up 2 File Manger windows.

Try This—Right-click For a New Folder

- From the explorer right-click on unused space inside a folder in which you want to create a new folder.
- Choose New Folder.
-

Shortcuts

Shortcuts are an abstract but extremely powerful tool for increasing efficiency and are especially useful in a networked environment. A user can create a shortcut to any object (such as file, program, network folder, Control Panel tool, disk drive, and so on) in the Chicago UI and place it anywhere else in the UI or in an application. When this shortcut is opened the object that the shortcut is "pointing" to is opened. For example, a shortcut to "My network folder" could be created and dropped on my desktop. When the shortcut is opened, it actually opens my network folder which is out on some network server somewhere. Shortcuts are represented just like regular icons, except that in the lower left corner there is a small "jump" arrow, as shown in Figure 15.

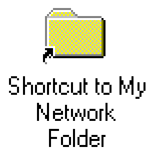


Figure 15. Chicago Shortcut Icon

A shortcut can be deleted without affecting the object to which it points. A shortcut can be created by selecting an object and choosing Create Shortcut from the File menu or from the right mouse click context menu. If shortcuts are created on an object that was created since Chicago was installed, then Chicago keeps track of renames. This means you can create a shortcut to \\Server\Share\Public Folder and put it on your desktop. Then if you or anyone else renames the network folder, the shortcut will still work regardless of the fact that the name of the folder it points to has changed. You can also rename shortcuts themselves.

Uses for shortcuts are virtually limitless, but some common powerful uses for shortcuts include:

- **Shortcuts in the Programs Folder**. Shortcuts are an extension of the concept behind the icons that used to appear in the Windows 3.1 Program Manager. They simply pointed to an .EXE file somewhere in the file system. In Chicago, the icons that appear in the Start Programs menu also appear as shortcuts in the Programs folder, which can be found by selecting Settings Programs from the Start Button. This way the user can keep shortcuts to all of his favorite programs in one central place, regardless of where the programs are actually installed. When a shortcut is added or deleted from the programs folder, likewise it is added or deleted from the Start Programs menu.
- **Shortcuts on the Desktop**. Power users will create shortcuts to commonly accessed files, programs, drives, folders, and utilities right on their desktops. This is especially powerful with network resources, because no complicated browsing or drive letter mapping is required to access network folders.
- **Embedded Shortcuts in applications**. For example, a user can drag a shortcut to a large file sitting on the network somewhere into a mail message. When the message recipient double clicks the shortcut the network file will be opened. This is much more efficient than embedding the actual file in a mail message because it is much smaller and it cuts down on version proliferation.

Try This—Discover Where the "Programs" Menu off the Start Button is Stored

Mouse

- Select Settings Programs... from the Start button.
- The Programs folder will open up (it is a sub-folder of the Windows folder)\

- Note all of the shortcuts and folders are exactly what appears in the Programs menu off the Start button. You can add or delete shortcuts and folders. This will change the items that appear in the Programs menu off the Start button.

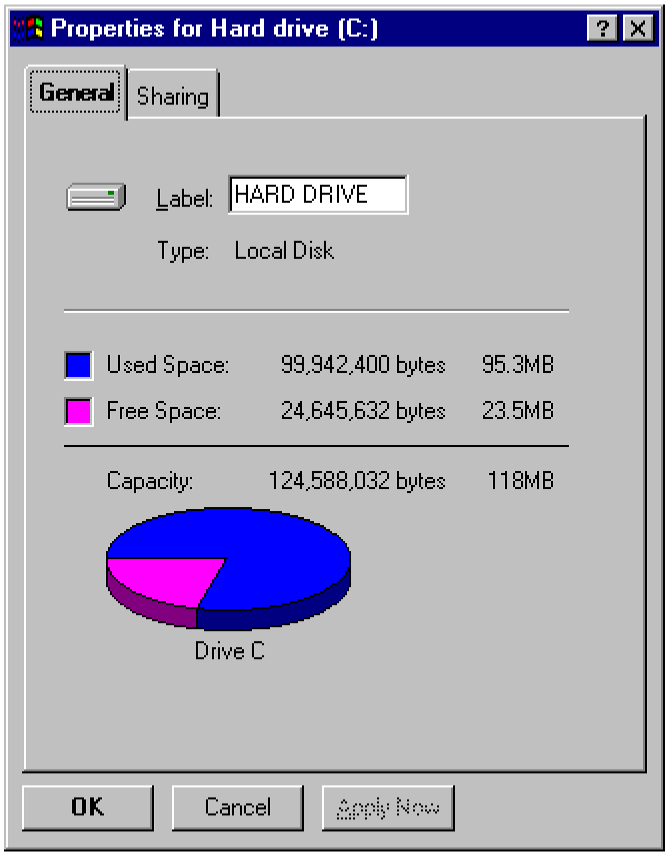
Properties Everywhere

Property sheets are an all-pervasive feature in Chicago. All objects in the UI carry context sensitive properties that can be accessed and customized by selecting File Properties or by right-clicking. Good, consistent, easily accessible properties sheets have been a favorite of power user testers to date. Properties will be illustrated through a series of "Try This" tips.

Try This—Rename Your Hard Drive in Disk Properties

Mouse

- From the Explorer or My Computer" right-click to select your hard disk.
- Choose Properties.
- Type a new name in the "Label" box. Choose OK.
- Choose View Refresh.
-



Label: HARD DRIVE

Type: Local Disk

🟦	Used Space:	99,942,400 bytes	95.3MB
🟪	Free Space:	24,645,632 bytes	23.5MB
	Capacity:	124,588,032 bytes	118MB

Figure 16. Chicago drive properties

Try This—Share a Folder

Mouse

- From the Explorer right-click select a folder you wish to make available to others on your network. Choose Properties.
- Select the Sharing tab.
- Select Shared As, then complete the other fields in this dialog box.
-

Try This—Customize a Shortcut Icon

Mouse

- Choose Start, Settings, Programs.
- Right-click on any shortcut. Choose Properties.
- Select the Shortcut tab.
- Click the Change Icon button.
- Select a new icon for the shortcut and choose OK.
- Select Refresh from the View menu.
-

Right-Clicking Everywhere

Right-clicking, like properties, is another all pervasive, context-sensitive feature of Chicago. (*Right-clicking* refers to clicking the secondary mouse button because most right-handed people set their mouse options to use the left button as primary and the right as secondary.) Usability tests have shown that in general, right-clicking is not a feature that novices discover or remember, therefore, the vast majority of functions performed on the right-click can also be performed by selecting the corresponding menu commands. However, right-clicking as a short cut for the most common actions to perform on an object has proven to be another very popular power user feature. The power of right-clicking is best illustrated through a series of "Try This" tips.

Try This—Right-click the Desktop to Customize

Mouse

- Right-click blank space on the desktop.
- Choose Properties.
-

Try This—Minimize All and Tile

- Right-click a blank place on the Windows Taskbar.
- Choose Minimize All or Tile Horizontally.
-

Try This—A Shortcut to Creating a Shortcut

- Right-click an object to which you would like to create a Shortcut.
- Choose Create Shortcut.
-

Try This—Non-default Drag and Drop

- Right-click and drag a file from the Explorer onto the desktop.

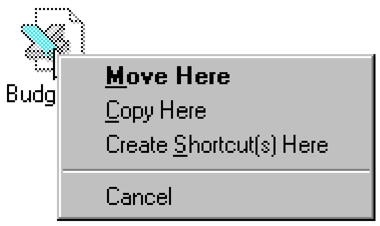


Figure 17. Non-default (right mouse button) drag and drop

Try This—Right-click a Screensaver to Install It

- Choose Find Files or Folders from the Start Button.
- Type "**bezier**" and choose Find Now.
- Right-click on Bezier.
- Choose Test.
-

Try This—Close a Task Right From The Taskbar

- Right-click on a Taskbar button.
- Choose Close.
-

Try This—Access Properties On an Open Window

- Right-click the mini-icon in the upper left hand corner of any open window.
- Choose Properties.
-

Control Panel: The Consolidated Control Center

The objective of the Control Panel special folder in Chicago is to consolidate into one location, all command, control, and configuration functions. A problem with Windows 3.1 was that these functions were difficult to find, use, and remember (such as, Windows Setup to change video resolution). The UI team has striven to create distinct and memorable visuals for all important functions and offer previews where appropriate. Individual Control Panel tool functionality will be covered in the section to which it pertains (such as, "Network" in the "Chicago networking and Systems Management" section of this guide).

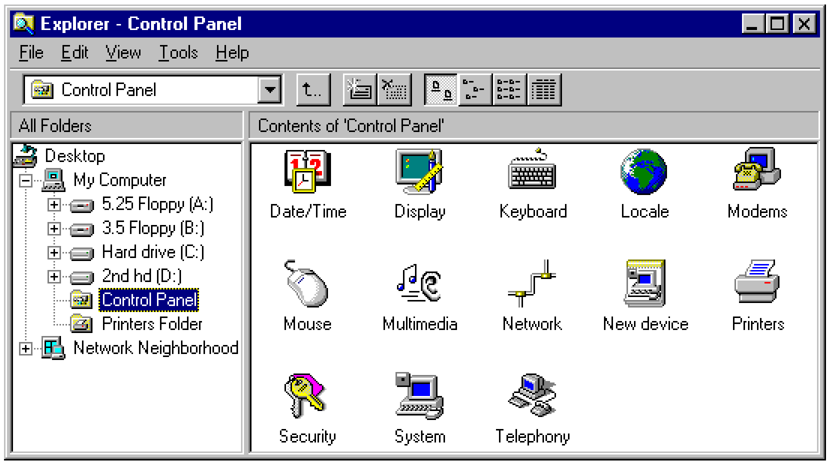


Figure 18. Explorer larger icon view of the Control Panel

There is one Control Panel tool, however, that pertains to customization of the UI itself, "Display". Display gives the user total control over the configuration of the Chicago UI allowing for personalization. Its four tabs are:

- **Background**. Allows pattern and wallpaper configuration and preview.
- **Screen Saver**. Allows screen saver configuration and preview.
- **Appearance.**: Allows configuration and preview of all of Chicago's UI metrics (fonts, sizes, colors, and so on).
- **Settings**. Allows configuration of monitor resolution and color palette size.

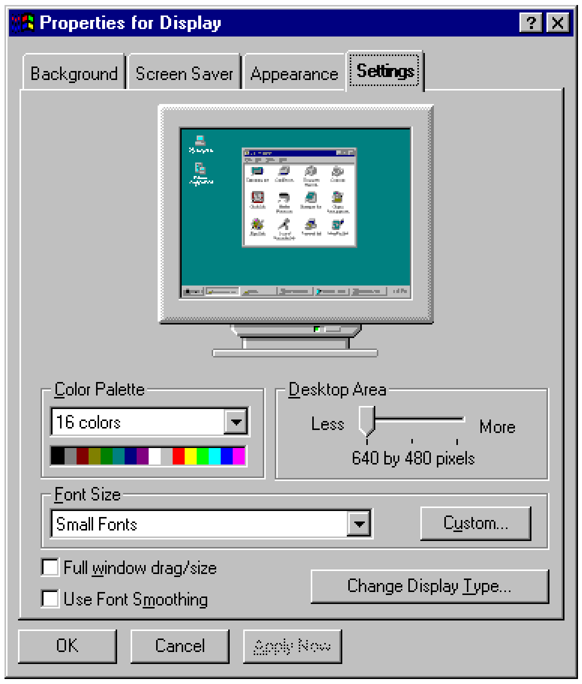


Figure 19. Chicago Display Properties

Try This—Full Screen Drag

(This "Try This" is only for 486/66 computers or better with 1MB+ video RAM.)

Mouse

- First, drag any window by grabbing the title bar with the mouse and moving it around. Note that you move around an outline of the window until you let go of the left mouse button.
- Now, right-click a blank area on the desktop.
- Choose Properties.
- Choose the Settings tab.
- Choose Full Screen Drag option from the lower left of the dialog. Chose OK.
- Now try dragging a window by grabbing the Title Bar. Notice that the window contents updates as you move.
-

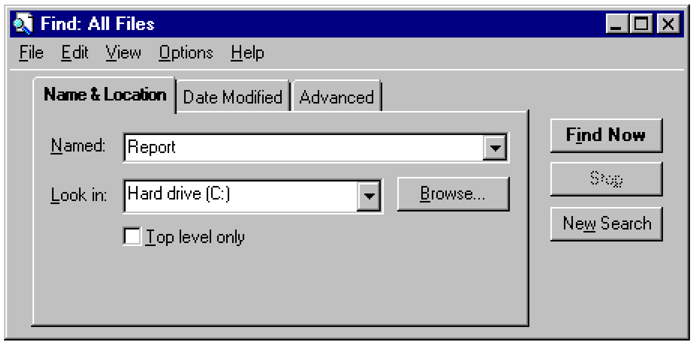
Try This—Hot Resolution Switch

Hot resolution switching is a feature that depends on several factors including type of video card. It is also work-in-progress, so it may not work in some cases.

- Choose Settings, Control Panel from the Start button.
- Open the Display tool.
- Choose the Settings tab.

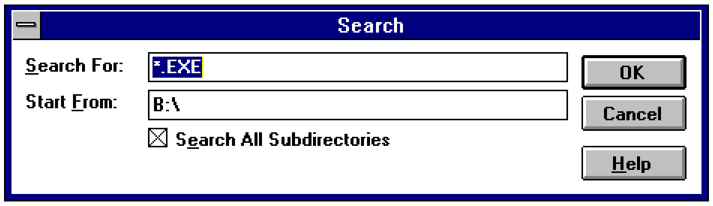
- Choose a different video resolution that is supported by your card by sliding the "Desktop Area" slider bar.
- Choose "Apply Now".
- If you are changing from 640 x 480 to a higher resolution, you may require a restart. However, after the initial restart, you should be able to perform hot switches between 800 x 600 and 1024 x 768 without restarting.
- Now try it with an AVI clip playing.
-

Find Files or Folders: Easily Location



Named: Report
Look in: Hard drive (C:)
☐ Top level only

Figure 20. Chicago's Find Files or Folders



Search For: *.EXE
Start From: B:\
☒ Search All Subdirectories

Figure 21. Search in Windows 3.1

A powerful new Find utility is built into Chicago. It goes far beyond the minimal functionality of the File Manger's Search utility in Windows 3.1. Features include:

- **Partial name searches**. Type "**rep**" in the Find Files Named window and all files and folders with rep in the name will be found.

- **Search on "Last Modification Date"**. Files can be searched on last modification date. Therefore, the user can perform searches like: "Find all Word documents modified in the last 3 days".

- **Full text search**. Full text of documents can also be searched.

- **Search results save**. Complex or useful searches can be saved.
- **File management from search results pane**. Rename files or look at file properties all from within the results pane just as if the user were in the Explorer.

Try This—Save a complex Find right on the Desktop

- From Start button select Find, then Files or Folders.
- Type a partial string that you know will be present in many files (such as, "**rep**" or "**doc**").
- Select the Date Modified tab.
- Choose modified during the previous seven days.
- Select the Advanced tab.
- Choose a file type if you want to.
- Push Find Now button.
- When find is complete choose Save Search from the File menu (notice, that because Find is 32-bit preemptively multi-tasked, while the Find is running you have control and can go perform other tasks).
- A Find Results icon is automatically created on the desktop. Double click it.
-

Printers Folder: Consolidated Printer Control

The Chicago Printers Folder offers one stop shopping for printer management and configuration. It replaces the troublesome Print Manger and Printers Control Panel Tool from Windows 3.1.

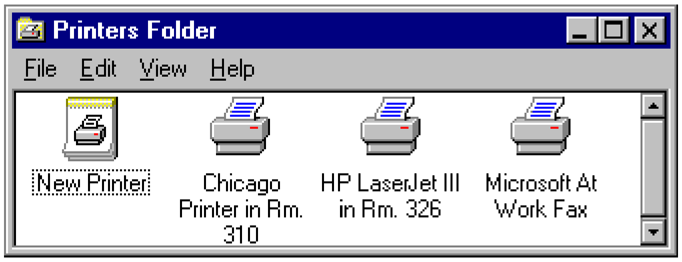
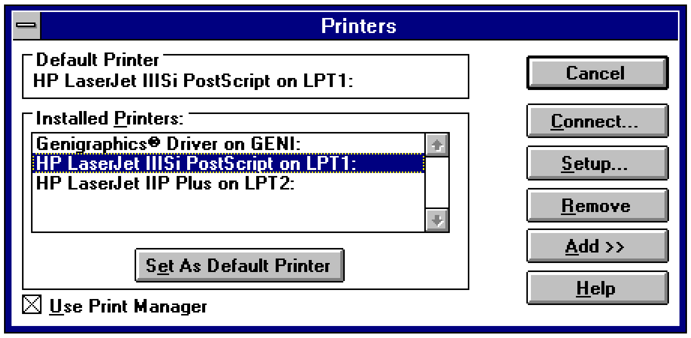


Figure 22. Chicago Printers Folder



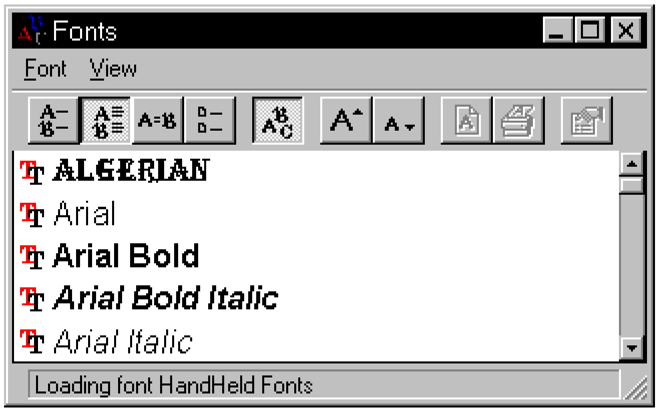
Default Printer
HP LaserJet IIISi PostScript on LPT1:

Installed Printers:
Genigraphics® Driver on GENI:
HP LaserJet IIISi PostScript on LPT1:
HP LaserJet IIP Plus on LPT2:

☒ Use Print Manager

Figure 23. Printer Configuration from Windows 3.1

Font Settings: More Powerful Font Management and Preview



ALGERIAN
Arial
Arial Bold
Arial Bold Italic
Arial Italic

Loading font HandHeld Fonts

Figure 24. Chicago Fonts Settings

New font management capabilities are only partially integrated into the UI in Beta-1. Going forward, the Fonts Folder will gain in power and accessibility. However, the Font Settings tool in Beta-1 is significantly improved over Windows 3.1.

Mouse

Try This—Preview Your Installed Fonts

- Select Settings Fonts from the Start button.
- Choose Display in Actual Font from the View menu.
- Select a font and choose Show Sample from the Fonts menu.
-

Quick Viewing of Files

Quick Viewers allow the user to get a preview of a file right from the UI, without having to open the application. Quick viewing of files is only partially implemented in Beta-1 and significant performance enhancements are planned. Chicago will include Quick Viewers for most typical file formats and will also evangelize ISVs to ship their own Quick Viewer drivers with future releases of their software.

Mouse

Try This—Quick-view a Bitmap

- Right-click on a Paint or Wordpad file.
- Choose Quick View .
-

Compatible

Compatibility is a requirement for Chicago. To be successful it must be a no-excuses, no-brainer upgrade from Windows 3.1. In the Chicago project overall, compatibility is most important for software and hardware when considering 3rd party sopftware and hardware. However, it also applies to the UI. The Chicago UI must be *compatible* with the way current Windows and MS-DOS users work today. The Chicago UI scales to the level and preferences of individual users.

Of primary importance, is that new UI features are easy for current Windows 3.1 users to learn at their own pace. While many UI compatibility features are present in Beta-1, a great deal of the UI modifications that will be made between Beta-1 and Beta 2 will focus on compatibility. Also, usability tests, which have focused on the new user to date, will shift focus to the problems faced by the upgrader. **Work continues...**

Program Manager and File Manager Run on Chicago

With minimal changes in appearance, the Program Manager and the File Manager run on Chicago and are easily accessible via the Start button. Several designs for access and default configuration for these Managers are underway. For example, when an upgrade boots into Chicago for the first time, the program manger might be opened as a window. Or, perhaps there will be a “Windows 3.1 compatibility” menu item on the Start button that will launch the Program Manager and the File Manger. Independent of the final design decision, there will be many help and learning devices (like the “Click Here to Begin” arrow that zooms across the Taskbar when you first boot into Chicago) that are specifically designed for the Upgrade.

MS-DOS Box and “Run” Command Get Better

For the “command-line junkie”, the usability and power of the MS-DOS Box have been dramatically improved. New capabilities include: Windows app launching, TrueType Fonts, Cut-Copy-Paste support, auto-sizing, and UNC pathname support.

The “Run” command, available off of the Start Button, has also been improved. It now supports UNC pathnames and has new browsing capabilities. Try it!

Improved ALT+TAB Task Switching

Despite the fact that task switching has been made dramatically easier and more accessible via the Chicago Taskbar, the familiar ALT+TAB “cool switch” has been updated. It now displays an iconic road map of all active tasks, to prevent getting lost in the infinite alt+tab loop that was common under Windows 3.1. Try it and see.

Base System Architecture

Ease on the surface requires power and speed at the core, and Chicago's modern, 32-bit architecture meets these requirements. Freed from the limitations of MS-DOS, Chicago preemptively multi-tasks for better PC responsiveness—so users will no longer have to wait while the system copies files, for example—and also delivers increased robustness and protection for applications. Chicago also provides the foundation for a new generation of easier, more powerful multi-threaded 32-bit applications. And most importantly, Chicago delivers this power and robustness on today's average PC platform while scaling well to take advantage of additional memory and CPU cycles.

The mission of Chicago is to deliver a complete, integrated, operating system, that offers modern 32-bit operating system technology, and includes built-in connectivity support. In addition to the high-level mission of Chicago, market requirements must be met to deliver a high performance, robust, and completely backwards-compatible operating system.

This section of the *Chicago Reviewer's Guide* discusses the base architecture used by Chicago. The base architecture covers low-level system services for managing memory, accessing disk devices, and providing robust support for running applications. Chicago delivers a modern 32-bit operating system that is compatible with existing software and hardware, and delivers a platform for new applications.

Summary of Improvements over Windows 3.1

Improvements made to the base architecture of Chicago result in many benefits to users. A summary of some of the key improvements include:

- Fully integrated 32-bit protected-mode operating system, eliminating the need for a separate copy of MS-DOS
- Preemptive multitasking, and multithreading support—improving system responsiveness and smooth background processing
- 32-bit installable file systems including VFAT, CDFS, and network redirectors supporting better performance, use of long filenames, and an open architecture supporting future growth
- 32-bit device drivers available throughout the system, delivering improved performance and intelligent memory use
- Complete 32-bit kernel, including memory management, scheduler, and process management
- Improved system-wide robustness and cleanup after an application ends or crashes, delivering a more stable and reliable operating environment

- More dynamic environment configuration reducing the need for users to tweak their system
- Improved system capacity, including better system resource limits to address issues Windows 3.1 users encountered when running multiple applications

Fully-Integrated Operating System

The first thing that users of Windows 3.1 and MS-DOS will see when they turn their computer on (or perhaps won't see) is the lack of an MS-DOS command prompt from which they would need to invoke Windows. Chicago is a tightly integrated operating system that features a preemptive multitasking kernel that boots directly into the graphical user interface, yet provides full compatibility with the MS-DOS operating system.

Many of Chicago's components overcome limitations inherent in MS-DOS and Windows 3.1, moreover, the improvements do not come at the cost of compatibility with existing software, hardware, or computing environment.

A Preemptive Multitasking Operating System

The job of the operating system is to provide services to the applications that are running in the system and, in a multitasking environment, to provide support for allowing more than one application to run concurrently. Windows 3.1 allowed multiple applications to run concurrently in the system in a *cooperative* multitasking manner. The Windows 3.1 operating system required an application to check the message queue every once in a while in order to allow the operating system to relinquish control to other running applications. Applications that did not check the message queue on a frequent basis would effectively hog all of the CPU time and prevent the user from switching to another running task.

Chicago uses a *preemptive* multitasking mechanism for running Win32–based applications and the operating system will take control away from or give control to another running task depending on the needs of the system. This means that unlike Win16–based applications, Win32–based applications do not need to *yield* to other running tasks in order to multitask in a friendly manner (Win16–based applications are still cooperatively multitasked for compatibility reasons). Chicago provides a mechanism for Win32–based applications to take advantage of the preemptive multitasking nature of the operating system to facilitate concurrent application design, called *multithreading*. A Win32–based application running in the system is called a *process* in terms of the operating system. Each process consists of at least a single *thread* of execution that identifies the code path flow as it is run by the operating system. A *thread* is a unit of code that can get a time slice from the operating system to run concurrently with other units of code, and must be associated with a process. However, a Win32–based application can *spawn* (or initiate) multiple threads for a given process to enhance the application for the user by improving throughput, enhancing responsiveness, and aiding background processing.

Due to the preemptive multitasking nature of Chicago, threads of execution will allow background code processing in a smooth manner.

For example, a word processing application (process) may implement multiple threads to enhance operation and simplify interaction with the user. The application may have a separate thread of code that responds to keys typed on the keyboard by the user to place characters in a document, while another thread is performing background operations such as spell-checking or pagination, while yet another thread is spooling a document to the printer in the background. Some Windows 3.1 applications that are available today may provide functionality similar to that just described, however because Windows 3.1 does not provide a mechanism for supporting multithreaded applications, it is up to the application vendor to implement their own threading scheme. The use of threads in Chicago facilitates application vendors to add asynchronous processing of information to their applications. Applications that use multithreading techniques in their applications will also be able to take advantage of improved processing performance available from Windows NT when using a symmetric multiprocessor (SMP) system by allowing different portions of the application code to run on different processors simultaneously (Windows NT uses a thread as the unit of code to schedule symmetrically among multiple processors).

Information about how Chicago runs MS-DOS–based applications in a preemptive manner (as Windows 3.1 does today), Win16–based applications in a cooperative manner (as Windows 3.1 does today), and Win32–based applications in a preemptive manner (as Windows NT does today), is provided later in this section.

No Need for CONFIG.SYS or AUTOEXEC.BAT

Chicago no longer needs a separate CONFIG.SYS or AUTOEXEC.BAT file as MS-DOS and Windows 3.1 require. Instead, Chicago is intelligent about the drivers and settings it needs to use and automatically will load the appropriate driver files or set the appropriate configuration settings during its boot process. If a CONFIG.SYS or AUTOEXEC.BAT file are present, the settings in these files will be used to set the global environment. For example, the default search path or the default appearance of the command prompt can be defined by using the appropriate entries in the AUTOEXEC.BAT file. While Chicago itself does not need a CONFIG.SYS or AUTOEXEC.BAT file, compatibility is maintained with existing software or environments that may require one or both of these files.

32-Bit Versus 16-Bit Components

Chicago uses a combination of 32-bit and 16-bit code in order to provide a good balance between delivering compatibility with existing applications and drivers, decreasing the size of the operating system working set, and offering improved system performance over Windows 3.1. System reliability is also improved without the cost of compatibility or increased size.

Chicago is a 32-bit preemptive multitasking operating system that implements some 16-bit code to provide compatibility with existing applications. In general, 32-bit code is provided in Chicago to maximize the performance of the system, while 16-bit code balances the requirements for reducing the size of the system and maintaining compatibility with existing applications and drivers.

Chicago's design deploys 32-bit code wherever it significantly improves performance without sacrificing application compatibility. Existing 16-bit code is retained where it is required to maintain compatibility, or where 32-bit code would increase memory requirements without significantly improving performance. All of the I/O subsystems and device drivers in Chicago, such as networking and file systems, are fully 32-bit, as are all the memory management and scheduling components (the kernel and virtual memory manager). Figure 25 depicts the relative distribution of 32-bit versus 16-bit code present in Chicago for system-level services. As can be seen from the figure, the lowest-level services provided by the operating system kernel are provided as 32-bit code. Most of the remaining 16-bit code consists of hand-tuned assembly language, delivering performance that rivals some 32-bit code used by other operating systems available on the market today.

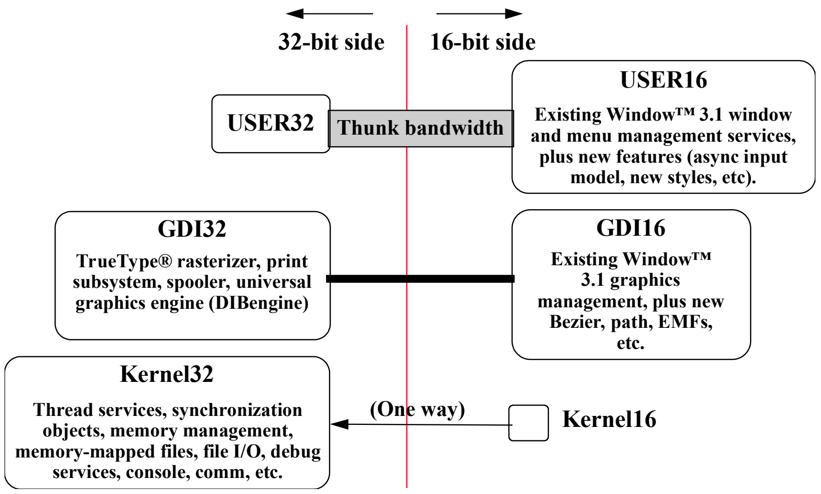

32-bit side
16-bit side
USER32
Thunk bandwidth
USER16
Existing Window™ 3.1 window and menu management services, plus new features (async input model, new styles, etc).
GDI32
TrueType® rasterizer, print subsystem, spooler, universal graphics engine (DIBengine)
GDI16
Existing Window™ 3.1 graphics management, plus new Bezier, path, EMFs, etc.
Kernel32
Thread services, synchronization objects, memory management, memory-mapped files, file I/O, debug services, console, comm, etc.
(One way)
Kernel16

Figure 25. Relative Code Distribution in Chicago

Many functions provided by the Graphics Device Interface (GDI) have been moved to 32-bit code, including the spooler and printing subsystem, the font rasterizer, and the drawing operations performed by the graphics "DIBengine." Much of the window management code (User) remains 16-bit to retain application compatibility.

In addition, Chicago improves upon the MS-DOS and Windows 3.1 environment by implementing many device drivers as 32-bit protected-mode code. Virtual device drivers in Chicago assume the functionality provided by many real-mode MS-DOS–based device drivers eliminating the need to load them in MS-DOS. This results in a minimal conventional memory footprint, improved performance, and improved reliability and stability of the system over MS-DOS–based device drivers.

Virtual Device Drivers—What is a VxD?

A virtual device driver (VxD) is a 32-bit, protected-mode driver that manages a system resource, such as a hardware device or installed software, so that more than one application can use the resource at the same time. To understand the improvements available in Chicago over the combination of MS-DOS and Windows 3.1, it is good to have a basic understanding of what a VxD is and the role virtual device drivers play in the Chicago environment.

The term *VxD* is used to refer to a general virtual device driver—the *x* represents the type of device driver. For example, a virtual device driver for a display device is known as a VDD, a virtual device driver for a timer device is a VTD, a virtual device driver for a printer device is a VPD, and so forth. Windows uses virtual devices to support multitasking for MS-DOS-based applications, virtualizing the different hardware components on the system to make it appear to each MS-DOS VM that it is executing on its own computer. Virtual devices work in conjunction with Windows to process interrupts and carry out I/O operations for a given application without disrupting how other applications run.

Virtual device drivers support all hardware devices for a typical computer, including the programmable interrupt controller (PIC), timer, direct-memory-access (DMA) device, disk controller, serial ports, parallel ports, keyboard device, math coprocessor, and display adapter. A virtual device driver can contain the device-specific code needed to carry out operations on the device. A virtual device driver is required for any hardware device that has settable operating modes or retains data over any period of time. In other words, if the state of the hardware device can be disrupted by switching between multiple applications, the device must have a corresponding virtual device. The virtual device keeps track of the state of the device for each application and ensures that the device is in the correct state whenever an application continues.

Although most virtual devices manage hardware, some manage only installed software, such as an MS-DOS device driver or a terminate-and-stay-resident (TSR) program. Such virtual devices often contain code that either emulates the software or ensures that the software uses data that applies only to the currently running application. ROM BIOS, MS-DOS, MS-DOS device drivers, and TSRs provide device-specific routines and operating system functions that applications use to indirectly access the hardware devices. Virtual device drivers are sometimes used to improve the performance of installed software; the 80386 and compatible microprocessors can run the 32-bit protected-mode code of a virtual device more efficiently than the 16-bit real-mode code of an MS-DOS device driver or TSR. In addition, performance is also enhanced by eliminating ring transitions that result in executing 32-bit applications that access 16-bit real-mode services—with virtual device drivers, the system can stay in protected-mode.

Chicago benefits from providing more device driver support implemented as a series of VxDs in the Windows environment, over the use of device drivers previously available as real-mode MS-DOS device drivers. Functionality that was previously supported as MS-DOS device drivers, but are now supported as VxDs in Chicago includes components such as:

- MS-DOS FAT file system
- SmartDrive
- CD-ROM file system
- Network redirector, network server, and network transport protocols
- Mouse driver
- MS-DOS SHARE.EXE TSR
- Disk device drivers including support for SCSI devices

In Chicago, VxDs provide improved performance due to a 32-bit code path and eliminating or reducing the need to mode switch between real and protected-mode, reduced conventional memory footprint by providing device driver and TSR functionality as protected-mode components that reside in extended memory, and improved system stability and reliability over using the MS-DOS device driver counterparts. Virtual device drivers can be identified by the use of a .VXD extension in Chicago, or a .386 extension as a virtual device driver from Windows 3.1.

Chicago System Layout

Figure 26 illustrates the layout of the base system architecture for Chicago. Components of the system are divided between Ring 0 and Ring 3 code, offering different levels of system protection. The Ring 3 code is protected from other running processes by protection services provided by the Intel processor architecture. The Ring 0 code consists of the low-level operating system services such as the file system, and virtual machine manager.

This figure also depicts the way that MS-DOS–, Win16–, and Win32–based applications run in the system. The following areas of this section discuss the provisions that the system makes for running these applications.

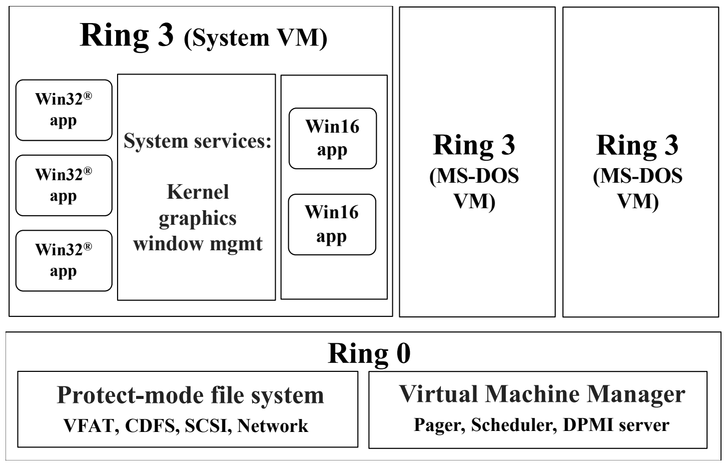


Figure 26. Chicago's Integrated Architecture for Running MS-DOS–, Win16–, and Win32–based Applications

Support for Win16–based Applications

16-bit Windows–based applications (Win16) run together within a unified address space, and are run in a cooperatively multitasking fashion as they do under Windows 3.1. Win16–based applications benefit from the preemptive multitasking of other system components including the 32-bit print and communications subsystem, and the improvements made in system robustness and protection from the Chicago system kernel.

When Win16–based application support was examined by the Chicago development team, three goals drove the architectural design based on customer needs, resource needs, and market needs: compatibility, size, and performance. Functionality such as running Win16–based applications together in the Win16 subsystem preemptively or running Win16–based applications in separate VMs was examined, however each option examined failed to meet the design goals set forth. The following discussion will provide some insight as to the architecture design of Chicago for running Win16–based applications in a fast, stable, and reliable way.

Compatibility

First and foremost, Chicago needs to run existing Win16–based applications without modification. This is extremely important to existing customers that want to take advantage of new functionality offered in Chicago such as 32-bit networking, but don't want to have to wait until new Chicago-enabled applications are available on the market.

Chicago builds upon the Windows 3.1 platform to provide support for running existing Win16–based applications and using existing Windows–based device drivers, while providing support for the next generation of 32-bit applications and components. Chicago extends the Windows 3.1 architecture in areas that have little or no impact on compatibility, as well as enhances the architecture to deliver a faster, more powerful 32-bit operating system.

Size

While many newer computer purchases are Intel 80486-based computers with 4MB or 8MB (or more) of memory, there are still a high percentage of 80386DX-based computers with 4MB of memory in use running Windows 3.1 today. To support the needs of the market, Chicago needs to run on a base platform of an Intel 80386DX-based computer with 4MB of RAM, to provide access to the new features and functionality provided, without requiring an upgrade of existing hardware or the addition of more RAM.

To meet its design goals, the Chicago development team designed Chicago to occupy no more working set than Windows 3.1 currently does, thereby insuring that any Win16–based application running at a perceived speed on a 4MB or 8MB computer (or greater) still runs at the same (or higher) speed under Chicago and does not suffer any performance degradation. To meet the required size goals of Chicago, Win16–based applications run within a unified address space, resulting in little overhead beyond that required by Windows 3.1 to support running Windows–based applications. This allows Chicago to not only simply fit on a 4MB computer, but also to perform well. The Chicago architecture includes innovative design features such as dynamically loadable VxDs to decrease the working set of components and memory requirements used by the operating system.

Meeting the size design goal (as well as to meet the compatibility goal), precluded the development team from adopting a strategy of running Win16–based applications in a separate VM by running a separate copy of Windows 3.1 on top of the operating system (thereby paying a several megabyte “memory tax” for each application) as OS/2 does, or emulating Windows 3.1 on top of the Win32 subsystem (thereby paying a "memory tax" for running Win16–based applications) as Windows NT does.

Running Win16–based applications in separate VMs is very expensive memory wise. This would require separate GDI, USER, and KERNEL code in each VM that is created, requiring the working set to increase by as much as 2MB for each Win16–based application that is running (as is required by OS/2 for Windows). If you have a computer with 16MB or more, this may not appear to be such a big deal. However, given the existing installed base of computers it would be impossible to run Win16–based applications in their own separate VMs in 4MB at all, and very difficult to run them in 8MB with the same level of performance as customers observe and expect under Windows 3.1 today.

Performance

Users expect their existing Win16 applications to run as fast or faster than they do under Windows 3.1. Win16–based applications will benefit from the 32-bit architecture of Chicago including the increased use of 32-bit device driver components and 32-bit subsystems, as will MS-DOS–based applications.

Win16–based applications run within a unified address space and interact with the system much as they do under Windows 3.1 today. Running Win16–based applications in separate VMs requires either a mapping of Win16 system components in each address space, as Windows NT does, or providing a separate copy of each system component in each address space, as OS/2 for Windows does. The additional memory overhead required for Win16 system components in each VM to run a Win16–based application has a negative impact on system performance.

Chicago balances the issue of system protection and robustness, with the desire for better system performance and improves on the system robustness over Windows 3.1. The improvements in this area are briefly discussed below, and are described in greater detail in a separate section of this guide.

Protection

The support for running Win16–based applications provides protection of the system from other running MS-DOS–based applications or Win32–based applications. Unlike Windows 3.1, an errant Win16–based application can not easily bring down the system or other running processes on the system. While Win32–based applications benefit the most from system memory protection, the robustness improvements present in Chicago result in a more stable and reliable operating environment than Windows 3.1.

Win16–based applications run within a unified address space, and cooperatively multitask as they do under Windows 3.1. The improvements made to overall system-wide robustness greatly enhance the system's ability to recover from an errant application, and lessens the likelihood of application errors due to improved clean up of the system. The occurrence of general protection faults (GPFs) under Windows 3.1 are most commonly caused by an application that writes over its own memory segments, rather than being caused by an application overwriting memory belonging to another application. Windows 3.1 did not recover gracefully when a Windows–based application crashed or hung. When an application was halted by the system due to a GPF, the system commonly left allocated resources in memory, causing the system to degenerate.

Due to improved protection in Chicago, an errant Win16–based application can not easily bring down either the system as a whole, or other running MS-DOS or Win32–based applications, and can at most impact other running Win16–based applications.

Other protection improvements include the use of separate message queues for each running Win32–based application. The use of a separate message queue for the Win16 address space and for each running Win32–based application provides better recovery of the system and doesn't halt the system should a Win16–based application hang.

Robustness Improvements

System robustness is also greatly improved when running Win16–based applications over Windows 3.1. Chicago now tracks resources allocated by Win16–based applications and uses the information to clean up the system after an application exits or ends abnormally, thus freeing up unused resources for use by the rest of the system.

Robustness improvements is discussed later in a separate section of this guide.

Support for MS-DOS–based Applications

There are many improvements in Chicago for running MS-DOS–based applications over Windows 3.1. As with Windows 3.1, each MS-DOS–based application runs in its own "virtual machine" (VM). A VM takes advantage of the Intel 80386 (and higher) architecture allowing multiple 8086-compatible sessions to run on the CPU, allowing existing MS-DOS applications to run preemptively with the rest of the system. As with Windows 3.1, the use of virtual device drivers provide common regulated access to hardware resources, thereby making each application running in a virtual machine think it's running on its own individual computer, allowing applications not designed to multitask to run concurrently with other applications.

Chicago provides a flexible environment for running MS-DOS–based applications. Unlike Windows 3.1, where users sometimes needed to exit Windows in order to run MS-DOS–based applications that were either ill-behaved or required direct access to system resources. MS-DOS–based application compatibility is improved in Chicago so almost all MS-DOS–based applications should run under Chicago.

Protection

VMs are fully protected from one another, as well as from other applications running on the system. This prevents errant MS-DOS–based applications from being able to overwrite memory occupied or used by system components or other applications. If an MS-DOS–based application attempts to access memory outside of its address space, the system will notify the user and the MS-DOS–based application will be ended.

Robustness Improvements

System robustness is also greatly improved when running MS-DOS–based applications over Windows 3.1. Robustness is discussed later in a separate section of this guide.

Improved Support for Running MS-DOS–based Applications

Chicago provides much better support for running MS-DOS–based applications within the Windows environment than Windows 3.1.

A detailed discussion of the improvements made to running MS-DOS–based applications is discussed in the section “Improved Support for Running MS-DOS–based Applications” later in this guide.

Support for Win32–based Applications

Win32–based applications can fully exploit and benefit more from the design of the Chicago architecture. In addition, each Win32–based application runs in its own fully-protected, private address space. This prevents other Win32–based applications from crashing each other, crashing other running MS-DOS–based applications, crashing running Win16–based applications, or crashing the Chicago system as a whole.

Win32–based applications feature the following benefits over Win16–based applications in Chicago or under Windows 3.1:

- Preemptive multitasking
- 32-bit Win32 APIs
- Long filename support
- Separate message queues
- Flat address space
- Memory Protection

Preemptive Multitasking

Unlike the cooperative multitasking used by Win16–based applications under Windows 3.1, 32-bit Win32–based applications are preemptively multitasked in Chicago. The operating system kernel is responsible for scheduling the time allotted for running applications in the system, and support for preemptive multitasking results in smoother concurrent processing and prevents any one application from utilizing all system resources without permitting other tasks to run.

Win32–based applications can optionally implement threads to improve the granularity at which they multitask within the system. The use of threads by an application improves the interaction with the user and result in smoother multitasking operation.

Separate Message Queues

Under Windows 3.1, the system uses the point when an application checks the system message queue as the mechanism to pass control to another task, allowing that task to run in a cooperative manner. If an application doesn’t check the message queue on a regular basis, or the application hangs and thus prevents other applications from checking the message queue, the system will keep the other tasks in the system suspended until the errant application is ended.

Each Win32–based application has its own separate message queue and is thus not affected by the behavior of other running tasks on their own message queues. If a Win16–based application hangs, or if another running Win32–based application crashes, a Win32–based application will continue to run preemptively and will still be able to receive incoming messages or event notifications.

Message queues are discussed in more detail in the "Robustness Improvements" section of this guide.

Flat Address Space

Win32–based applications benefit from improved performance and simpler construct due to being able to access memory in a linear fashion, rather being limited to the segmented memory architecture used by MS-DOS and Windows 3.1. In order to provide a means of accessing high amounts of memory using a 16-bit addressing model, the Intel CPU architecture provides support for accessing 64K chunks of memory at a time, called segments. Applications and the operating system suffer a performance penalty under this architecture due to the necessary manipulations required by the processor for mapping memory references from the segment/offset combination to the physical memory structure.

The use of a flat address space by Chicago's 32-bit components and for Win32–based applications will allow application and device driver developers to write software without the limitations or design issues inherent with the segmented memory architecture used by MS-DOS and Windows 3.1.

Compatibility with Windows NT

Win32–based applications that exploit Win32 APIs common between Chicago and Windows NT can run without modification on either platform on Intel-based computers. The commonality of the Win32 API provides a consistent programmatic interface allowing application vendors to use a single development effort to leverage delivery of software that runs on multiple platforms. This provides scalability of applications and broadens the base of platforms available for running ISV or custom applications with minimal additional effort.

Application vendors are encouraged to develop applications either under Chicago or Windows NT, and test compatibility on both platforms.

Long Filename Support

Win32–based applications that call the file I/O functions supported by the Win32 API will benefit from the ability to support and manipulate filenames up to 255 characters, with no additional development effort. The Win32 APIs and common dialog support handles the work for manipulating long filenames, and the file system provides compatibility with MS-DOS and other systems by also maintaining the traditional 8.3 filename automatically. This eases the burden from the application developer.

Memory Protection

Each Win32–based application runs in its own private address, and is protected by the system from other applications or processes that are running in the system. Unlike running Win16–based applications under Windows 3.1, errant Win32–based applications under Chicago will only end themselves, rather than bring down the entire system if they attempt to access memory belonging to another application.

The use of separate message queues for Win32–based applications also protects to ensure that the system will continue to run if an application hangs or stops responding to messages or events.

Robustness Improvements

Win32–based applications benefit from the highest level of system robustness supported under Chicago. Resources allocated for each Win32–based application is tracked on a per-thread basis and are automatically freed when the application ends. If an application hangs, users are able to perform a *local reboot* operation to end the hung application without affecting other running tasks, and the system will clean up properly.

Detailed information about robustness enhancements is discussed later in a separate section of this guide.

32-Bit File System Architecture

The file system in Chicago has been re-architected from Windows 3.1 to support the characteristics and needs of the multitasking nature of the Chicago kernel. The changes present in Chicago provide many benefits to the user and results in:

- **Improved ease of use**

  Ease of use is improved by support long filenames so users no longer need to reference files by the MS-DOS 8.3 filename structure—users can use up to 255 characters to identify their documents. Ease of use is also improved by hiding the filename extensions from users.

- **Improved performance**

  As in Windows for Workgroups 3.11, file I/O performance is improved dramatically over Windows 3.1 by featuring 32-bit protected-mode code for reading information from and writing information to the file system, reading and writing information from/to the disk device, and intelligent 32-bit caching mechanisms—a full 32-bit code path is available from the file system to the disk device.

- **Improved system stability and reliability**

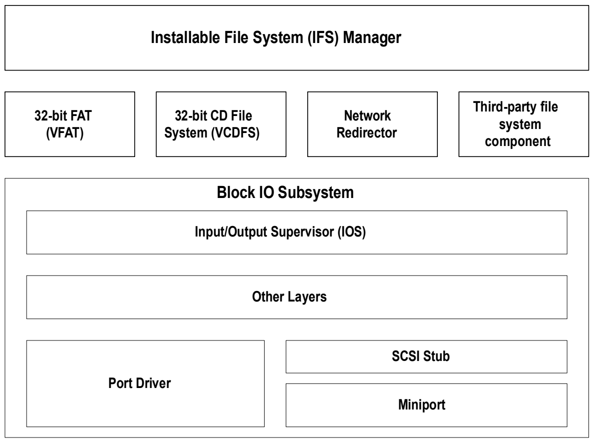
  File system components implemented as 32-bit protected mode device drivers offer improved system stability and reliability over MS-DOS device driver counterparts due to being able to remain in protected-mode for code execution and leveraging existing driver technology first implemented in Windows NT and also available in Windows for Workgroups 3.11.

Architecture Overview

Chicago features a layered file system architecture that supports multiple file systems, and provides a protected-mode path from the application to the media device, resulting in improved file and disk I/O performance over Windows 3.1. Features of the new file system architecture include:

- Win32 API support
- Long filename support
- 32-bit FAT file system
- 32-bit CD-ROM file System
- Dynamic system cache for file and network I/O
- Open architecture for future system support
- Disk device driver compatibility with Windows NT

Figure 27 depicts the file system architecture used by Chicago.



Installable File System (IFS) Manager
32-bit FAT (VFAT)
32-bit CD File System (VCDFS)
Network Redirector
Third-party file system component
Block IO Subsystem
Input/Output Supervisor (IOS)
Other Layers
Port Driver
SCSI Stub
Miniport

Figure 27. Chicago File System Architecture

The Chicago file system architecture is made up of the following components:

- **Installable File System (IFS) Manager**

  The IFS Manager is responsible for arbitrating access to different file system components.

- **File system drivers**

The file system drivers layer includes access to file allocation table (FAT)-based disk devices, CD-ROM file systems, and redirected network device support.

- **Block I/O subsystem**

  The block I/O subsystem is responsible for interacting with the physical disk device.

We'll examine components of each of these layers in this section.

Installable File System Manager

Under MS-DOS and Windows 3.1, the MS-DOS Int 21h interrupt is responsible for providing access to the file system to manipulate file information on a disk device. In order to support redirected disk devices (for example, a network drive, or a CD-ROM drive), other system components such as the network redirector would hook the Int 21h function so that it could examine the file system request to determine whether it should handle the file I/O request, or let the base file system handle it. While this mechanism provided the ability to add on additional device drivers, some add-on components would be ill-behaved and would interfere with other installed drivers.

Another problem that was encountered with the MS-DOS–based file system was the difficulty in supporting the loading of multiple network redirectors to provide concurrent access to different network types. Windows for Workgroups provided support for running the Microsoft Windows Network redirector at the same time as an additional network redirector including Novell NetWare, Banyan VINES, and SUN PC-NFS, however support for running more than two network redirectors at the same time was not supported.

The key to friendly access to disk and redirected devices in Chicago is the Installable File System (IFS) Manager. The IFS manager is responsible for arbitrating access to file system devices, as well as other file system device components.

The Chicago includes support for the following file systems:

- 32-bit FAT driver (VFAT)
- 32-bit CD-ROM file system driver (CDFS), and
- 32-bit network redirector for connectivity to Microsoft Windows Network servers like Windows NT Advanced Server, along with a 32-bit network redirector to connect to Novell NetWare servers

Third-parties will use the IFS Manager APIs to provide a clean way of concurrently supporting multiple device types, adding additional disk device support and network redirector support.

32-bit File Access—Protected-mode FAT (VFAT) File System

The 32-bit VFAT driver provides a 32-bit protected-mode code path for manipulating the file system stored on a disk. It is also re-entrant and multi-threaded, providing smoother multi-tasking performance. The 32-bit file access driver is improved over that provided originally with Windows for Workgroups 3.11, and is compatible with more MS-DOS-device drivers and hard disk controllers.

Benefits of the 32-bit file access driver over MS-DOS–based driver solutions include:

- Dramatically improved performance and real-mode disk caching software
- No conventional memory used—replacement for real-mode SmartDrive
- Better multitasking when accessing information on disk—no blocking
- Dynamic cache support

Under MS-DOS and Windows 3.1, manipulation of the file allocation table (FAT) and writing or reading information to/from the disk is handled by the Int 21h MS-DOS function and is 16-bit real-mode code. Being able to manipulate the disk file system from protected-mode removes or reduces the need to transition to real-mode in order to write information to the disk through MS-DOS, which will result in a performance gain for file I/O access.

The 32-bit VFAT driver interacts with the block I/O subsystem to provide 32-bit disk access to more device types than is supported by Windows 3.1. Support is also provided for mapping to existing real-mode disk drivers that may be in use on a user's system. The combination of the 32-bit file access and 32-bit disk access drivers result in significantly improved disk and file I/O performance.

32-Bit Cache—VCACHE

The 32-bit VFAT works in conjunction with a 32-bit protected-mode cache driver (VCACHE), and replaces and improves on the 16-bit real-mode SmartDrive disk cache software provided with MS-DOS and Windows 3.1. The VCACHE driver features more intelligent caching algorithm than SmartDrive to cache information read from or written to a disk drive, and results in improved performance for reading information from cache. Also, the VCACHE driver is responsible for managing the cache pool for the CD-ROM File System (CDFS), and the provided 32-bit network redirectors.

Another big improvement in VCACHE over SmartDrive is that the memory pool used for the cache is *dynamic* and is based on the amount of available free system memory. Users no longer need to statically allocate a block of memory to set aside as a disk cache, the system automatically allocates or de-allocates memory used for the cache based on system use. The performance of the system will also scale better than Windows 3.1 or Windows for Workgroups 3.11, due to the intelligent cache use.

32-Bit CDFS—Protected-mode CD-ROM File System

The 32-bit protected-mode CD-ROM file system (CDFS) implemented in Chicago provides improved CD-ROM access performance over the real-mode MSCDEX driver in Windows 3.1 and is a full 32-bit ISO 9660 CD file system. The CDFS driver replaces the 16-bit real-mode MSCDEX driver, and features 32-bit protected-mode caching of CD-ROM data. The CDFS driver cache is dynamic and shares the cache memory pool with the 32-bit VFAT driver, requiring no configuration or static allocation on the part of the user.

Benefits of the new 32-bit CDFS driver include:

- No conventional memory used—replacement for real-mode MSCDEX
- Improved performance over MS-DOS–based MSCDEX and real-mode cache
- Better multitasking when accessing CD-ROM information—no blocking
- Dynamic cache support to provide a better balance between providing memory to run applications versus memory to serve as a disk cache

If MSCDEX is specified in the user's AUTOEXEC.BAT, the 32-bit CDFS driver will take over role played by the MSCDEX driver and communicate with the CD-ROM device. The use of MSCDEX is no longer necessary under Chicago.

Users of CD-ROM multimedia applications will benefit greatly from the new 32-bit CDFS. Their multimedia applications will run smoother and information will be read from the CD-ROM quicker providing improved performance.

Disk Device Architecture—Block I/O Subsystem

The Block I/O Subsystem in Chicago improves upon the 32-bit disk access "FastDisk" device architecture used in Windows 3.1 to improved performance for the entire file system and a broader array of device support.

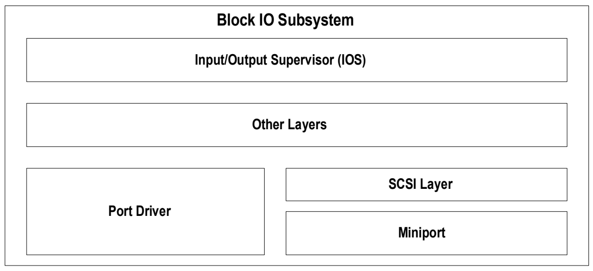


Figure 28. Architecture of Chicago Block I/O Subsystem

Components of the block I/O subsystem include the high-level I/O Supervisor (IOS) layer, which provides the interface to the block I/O subsystem to the higher layer components; the port driver, which represents a monolithic disk device driver; the SCSI layer, which provides a standard interface and driver layer to provide device-independent control code for SCSI devices; and the SCSI mini-port driver, which contains the device-dependent control code responsible for interacting with individual SCSI controllers.

The block I/O subsystem provides the following support in Chicago:

- Fully Plug and Play-enabled architecture
- Support for mini-port drivers that are binary compatible with Windows NT
- Support for Windows 3.1 fast disk drivers for backwards compatibility
- Protected-mode drivers that take over real-mode MS-DOS device drivers if it is thought to be safe to do so
- The ability to support existing MS-DOS real-mode disk device drivers for compatibility

Let's examine the different areas that make up the block I/O subsystem. Keep in mind that the configuration of the disk device driver layers is isolated from the user, so the explanation here is provided to facilitate an understanding of the components.

I/O Supervisor

The I/O Supervisor (IOS) provides services to file systems and drivers. The IOS is responsible for the queuing of file service requests and for routing the requests to the appropriate file system driver. The IOS also provides asynchronous notification of file system events to drivers that are installed.

Port Driver

The port driver is a monolithic 32-bit protected-mode driver that communicates with a specific disk device such as a hard disk controller. This driver is Chicago-specific and resembles the 32-bit disk access (fast disk) driver used in Windows 3.1 (for example, WDCTRL for Western Digital compatible hard disk controllers). In Chicago, the driver to communicate with IDE/ESDI hard disk controllers and floppy disk controllers is implemented as a port driver. A port driver provides the same functionality as the combination of the SCSI manager and the mini-port driver.

SCSI Layer

The SCSI layer applies a 32-bit protected-mode universal driver model architecture to communicating with SCSI devices. The SCSI layer provides all the high level functionality that is common to SCSI-like devices, and then uses a mini-port driver to handle device-specific I/O calls. The SCSI Manager is also part of this system and provides the compatibility support for using Windows NT mini-port drivers.

Mini-Port Driver

The Chicago mini-port driver model simplifies the task for a hardware disk device vendor to write a device driver. Because the SCSI Stub provides the high level functionality for communicating with SCSI devices, the hardware disk device vendor only needs to create a mini-port driver that is tailored to his own disk device. The Chicago mini-port driver is 32-bit protected-mode code, and is binary compatible with Windows NT mini-port drivers, minimizing the task required by a hardware vendor to write device drivers. Binary compatibility with NT also results in a more stable and reliable device driver as the hardware vendor needs to only maintain one code base for device support, and Chicago users benefit from the preexistence of many mini-port drivers already available for Windows NT.

Support for IDE, SCSI, ESDI controllers

Through the use of either a port driver, or a mini-port driver, support for a broad array of disk devices will be available when Chicago ships including popular IDE, ESDI, and SCSI disk controllers. Keep in mind that users don't have to decide whether to use a port driver or a mini-port driver, the driver is provided by the hardware vendor and configuration of the driver is handled by the Chicago system.

Real-Mode Mapper (RMM)

To provide compatibility with real-mode MS-DOS device drivers for which a protected-mode counterpart does not exist, the block I/O subsystem provides a mapping layer to allow the protected-mode file system to communicate with a real-mode driver as if it was a protected-mode component. The layers above and including the real-mode mapper are protected-mode code, and the real-mode mapper translates file I/O requests from protected-mode to real-mode such that the MS-DOS device driver can perform the desired operation to write or read information to or from the disk device. An example scenario where the real-mode mapper would come into play is when real-mode disk compression software is running and a protected-mode disk compression driver is not available. The net effect of this component is to ensure binary compatibility with existing MS-DOS–based disk device drivers in Chicago.

Long Filename Support

The use of long filenames in Chicago overcomes the sometimes cryptic 8.3 MS-DOS filename conventions, to allow more user friendly filenames. MS-DOS 8.3 filenames are still maintained and tracked by the system to support compatibility with existing Win16 and MS-DOS–based applications that only manipulate 8.3 filenames, but as users migrate to Win32–based applications the use of 8.3 filename conventions is hidden from the user. Long filenames can be up to 255 characters in length.

Long filenames are supported by extending the MS-DOS FAT file system and using bits and fields that were previously reserved by the operating system to add special directory entries that maintain long filename information. Extending the MS-DOS FAT layout, rather than creating a new format, allows users to install and use Chicago on existing disk formats without having to change their disk structure, or reformat their drives. This implementation provides future growth and ease of use, while still maintaining backward compatibility with existing applications.

Because Chicago simply extend the FAT structure, support for long filenames is support on diskettes as well as hard disk drives. If a long filename is used for a file on a diskette and is viewed by a user on a computer not running Chicago, the user would only see the 8.3 filename representations.

Figure 29 shows a disk directory on a Chicago computer showing long filenames and the corresponding 8.3 filename mappings.

```
Volume in drive C is MY HARDDISK
 Volume Serial Number is 1B47-7161
 Directory of C:\LONGFILE

.              <DIR>        05-11-94 10:34a .
..             <DIR>        05-11-94 10:34a ..
4THQUART XLS           147  05-11-94 12:25a 4th Quarter Analysis.xls
TEXTFILE TXT           147  05-11-94 12:25a TEXTFILE.TXT
THISISMY DOC           147  05-11-94 12:25a this is my long filename.doc
1994FINA DOC           147  05-11-94 10:35a 1994 Financial Projections.doc
         4 file(s)            588 bytes
         2 dir(s)      48,009,216 bytes free
```

Figure 29. Directory with Long Filenames Visible from Command Prompt

Support for Existing Disk Management Utilities

In order for existing disk management utilities to recognize and preserve long filenames, utility vendors will need to revise their software offerings. Microsoft is working closely with utilities vendors and is documenting long filename support and its implementation as an extension to the FAT format as part of the *Chicago SDK*.

Existing disk management utilities that manipulate the FAT, including disk defragmenters, disk bit editors, and some tape backup software, may not recognize long filenames as used by Chicago and may destroy the long filename entries in the FAT. However, the corresponding system-defined 8.3 filename will be preserved so there is no loss of data if the long filename entry is destroyed.

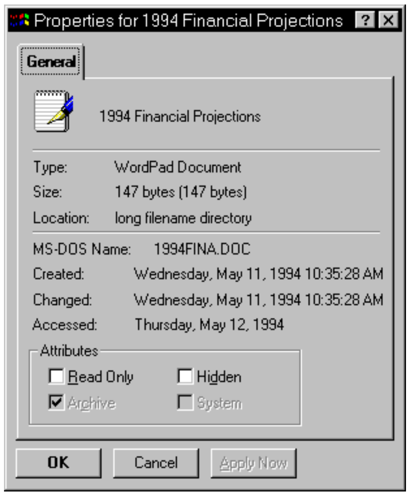
File Extensions Hidden From User

File extensions are used by Chicago to associate a given file type with an application as is handled under Windows 3.1. However, file extensions are hidden from users in the Shell and Explorer to make it easier to manipulate files, and icons are used in the Chicago UI to differentiate documents associate with applications. For compatibility reasons, it is still necessary for Chicago to track filename extensions for use with existing MS-DOS and Win16–based applications. Information on the file type associations is stored in the Registry, and the associations are used to map a given file with the appropriate icon representing the document type.

In addition to hiding filename extensions in the Chicago shell and Explorer, mechanisms are available for application developers to hide filenames from users in their applications, and this is documented in the *Chicago SDK*. A good Chicago application will make use of these mechanisms for handling files to be consistent with the rest of the Chicago environment.

Additional File Date/Time Attributes

To further enhance the file system, Chicago maintains additional date/time attributes for files that MS-DOS does not track. Chicago will now maintain the date/time when a new file is created, the date/time when a file has been modified or changed, and the date when a file was last opened. These file attributes will be displayed when a user requests to display file properties as shown in Figure 30.



1994 Financial Projections

Type:	WordPad Document
Size:	147 bytes (147 bytes)
Location:	long filename directory
MS-DOS Name:	1994FINA.DOC
Created:	Wednesday, May 11, 1994 10:35:28 AM
Changed:	Wednesday, May 11, 1994 10:35:28 AM
Accessed:	Thursday, May 12, 1994

Attributes
- ☐ Read Only
- ☐ Hidden
- ☑ Archive
- ☐ System

Figure 30. Properties for a File, Showing New File Attributes

Utilities can take advantage of this additional time/date information to provide enhanced backup utilities, for example, to use a better mechanism when determining whether a given file has been changed or modified by the system.

Coordinated Universal Time (UTC) Format

MS-DOS has traditionally used the local time of the computer as the time stamp for the directory entry of a file. Chicago will continue to do this for files stored on the local system, however support for using the UTC time format for accessing or creating information on network file servers. This will provide better, more universal tracking of time information as required by networks that operate across time zones.

Exclusive Volume Access For File Recover Tools

Today, disk management utilities such as disk defragmenters, sector editors, and disk compression utilities, don’t get along well with Windows 3.1. File system programs, such as CHKDSK and DEFRAG, require special (exclusive) access to the file system to minimize the disk access complexities that are present in a multi-tasking environment where disk I/O occurs. For example, if a user requests to do a disk operation that moves files or information around on the disk, if another task was accessing the information or writing information to disk at the same time, without exclusive access to the disk it would be possible that data corruption could occur. Windows 3.1 and MS-DOS do not provide a means of controlling access to the disk when other tasks may need to write information out at the same time, and it is for this reason that it is necessary today for users to exit Windows and enter MS-DOS to run disk management utilities.

The Chicago file system has been enhanced to permit exclusive access to a disk device to support the use of Windows–based disk management utilities. This is not an end-user feature, but rather is an end-user benefit. Exclusive disk access is handled through a new API mechanism as part of the file system and can be used by utilities' vendors to write Windows–based disk management utilities. Microsoft is evangelizing this API mechanism to third-party utility vendors to facilitate moving existing MS-DOS–based utilities to Windows, as well as is using it to deliver disk management utilities as part of the Chicago product.

For example, this mechanism is being used in Chicago by the disk defragment utility delivered as part of the Beta-1 release. Unlike the combination of MS-DOS and Windows 3.1, the disk defragment utility in Chicago can be run from the Chicago shell, *and* can even be run in the background while you continue to work on your system.

Improved System Capacity

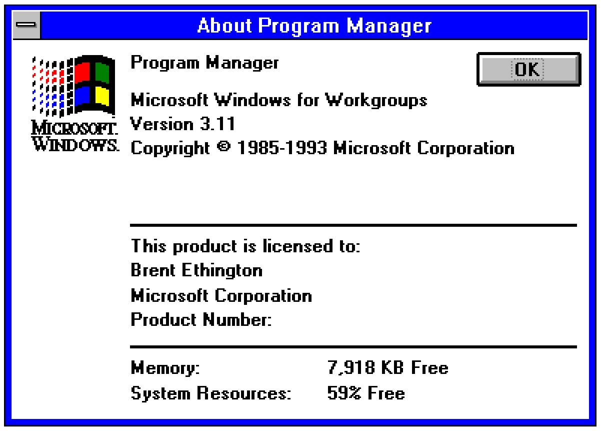
Chicago provides better system capacity for running MS-DOS-based and Win16-based applications than Windows 3.1. A number of internal enhancements have been made to the base system, allowing for internal system resources to not be exhausted as quickly as was possible under Windows 3.1 when running multiple Windows–based applications.

Many of the artificial limitations present in Windows 3.1 due to its architecture or internal data structures and largely due to the fact that Windows 3.1 had to run on an Intel 80286-based computer, have been greatly improved and overcome in Chicago. This will please ISVs and other developers, as well as end-users.

System Resource Limitation Improved

Many users have probably seen "Out of Memory" error messages when running multiple Windows–based applications under Windows 3.1, even though the system still reports several megabytes of available free memory. What users typically encountered was a condition where the system was not able to allocate an internal memory resource in a Windows API function call due to not enough space available in a region of memory called a *heap*.

Windows 3.1 maintains heaps for system components called GDI and USER. Each of the heaps is 64K in size and is used for storing GDI or memory *object* information allocated when an application calls a Windows API function. The amount of space available in the combination of these two heaps is identified as a percentage of system resources that are free and is shown in the Help About box in Program Manager and other Windows applications as shown in Figure 31.



Program Manager

Microsoft Windows for Workgroups
Version 3.11
Copyright © 1985-1993 Microsoft Corporation

This product is licensed to:
Brent Ethington
Microsoft Corporation
Product Number:

Memory:	7,918 KB Free
System Resources:	59% Free

Figure 31. About Box in Program Manager In Windows 3.1 Showing Free System Resources

The percentage of free system resources displayed in the About box is calculated using an internal algorithm to represent the aggregate percentage of free memory in the GDI and USER heaps. When the free system resources percentage drops to a low number, it is quite common that the user will see an "out of memory" error message, even though the amount of free memory shown in the About box is still quite high. This error can be due to low memory in either the GDI or the USER heap (or both).

To help reduce the system resource limitation, a number of the data structures stored in the 16-bit GDI and USER heaps in Windows 3.1 have been moved out of these heaps and stored in 32-bit heaps, providing more room for the remaining data elements to be created. Users will see improvements by not encountering a decrease in system resources as rapidly as they may have seen with Windows 3.1.

All objects were not simply removed from the 16-bit GDI or USER heaps, and placed in 32-bit heaps for compatibility reasons. For example, there are some Windows–based applications that manipulate the contents of the GDI heap directly, bypassing the published API mechanisms for doing so. These application vendors to do this for perceived performance reasons. However, because they bypass the Windows API mechanisms, moving the data from the existing heap structures and placing them in 32-bit heaps would cause the existing applications to fail due to memory access violations.

Both Win16 and Win32–based applications use the same GDI and USER heaps. The impact of removing selected items from the heaps was closely examined and objects were selected based on the biggest improvement that could be achieved, while affecting the fewest number of applications. For example, the GDI heap can quickly become full due to the creation of memory-intensive region objects that are used by applications for creating complex images and by the printing subsystem for generating complex output. Regions have been removed from the 64K 16-bit GDI heap and placed into a 32-bit heap, benefiting graphic-intensive applications and providing for the creation of more smaller objects by the system. Chicago improves the system capacity for the USER heap, by moving menu and window handles to the 32-bit USER heap, raising the total limit of these data structures from 200 in Windows 3.1, to a total limit now of 32,767 menu handles and an additional 32,767 window handles *per* process rather than system wide.

In addition to examining information present in the GDI and USER heaps, the robustness improvements present in Chicago that facilitate cleaning up the system of unfreed resources will also help the system resource limitation problem. Chicago will clean up and de-allocate left over data structures once Chicago determines that the owner and other ended processes no longer need the resources in memory. The robustness improvements available in Chicago are discussed in the next section.

Better Memory Management

Chicago improves addressibility for accessing physical memory in the computer, as well as improves upon the swapfile implementation provided in Windows 3.1 to support virtual memory to supplement physical system memory.

Linear Memory Addressing for Win32–based Applications

To support a 16-bit operating environment, the Intel processor architecture uses a mechanism called *segments* to reference memory by using a 16-bit segment address, and a 16-bit offset address within the segment. A segment is 64K in size, and applications and the operating system endure a performance penalty for accessing information across segments. Chicago addresses this issue by using the 32-bit capabilities of the Intel 80386 (and above) processor architecture to support a flat, linear memory model for 32-bit operating system functionality and Win32–based applications. A linear addressing model simplifies the development process for application vendors, removes the performance penalties imposed by the segmented memory architecture, and provides access to a virtual address space that enables addressing up to 4 gigabytes (GB) of memory. Chicago uses the flat memory model internally for 32-bit components and virtual device drivers.

Compatible with the Memory Model used by Windows NT

Chicago uses the same memory model architecture used by Windows NT, providing high-end operating system functionality on the mainstream desktop. Chicago will allow full use of the 4 gigabytes (4 billion bytes of memory) of addressable memory space to support even the largest desktop application.

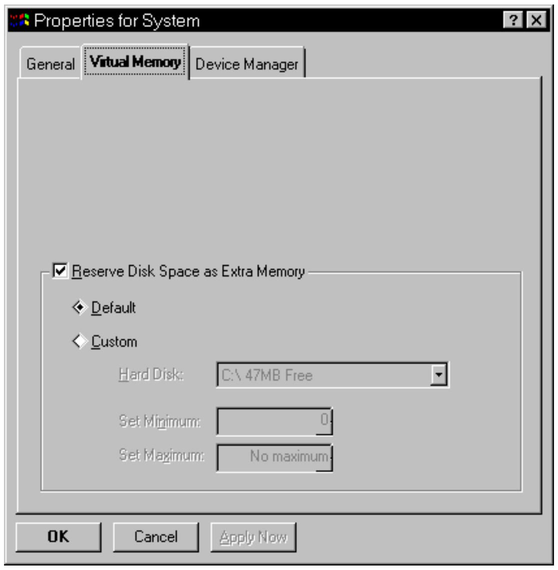
Improved Virtual Memory Support—Swapfile Improvements

Chicago improves on the virtual memory swapfile implementation provided in Windows 3.1 to address the problems and limitations imposed in Windows 3.1.

Under Windows 3.1, users were faced with a myriad of choices and configuration options when it came to setting up a swapfile to support virtual memory. They had to decide whether to use a temporary swapfile or a permanent swapfile, how much memory to allocate to the swapfile, and whether to use 32-bit disk access to access the swapfile or not. Users benefited from a temporary swapfile in that the swapfile did not need to be contiguous, and Windows would allocate space on the hard disk when Windows was started and free up the space when the user exited Windows. A permanent swapfile provided the best performance, however it required a contiguous block of space, had to be set up on a physical hard disk, and was statically specified by the user and not freed up when the user exited Windows.

The swapfile implementation in Chicago simplifies the configuration task for the user and combines the best of a temporary swapfile and a permanent swapfile due to improved virtual memory algorithms and access methods. The swapfile in Chicago is now dynamic, and can shrink or grow based on the operations that are performed on the system. The swapfile can also occupy a fragmented region of the hard disk, with no substantial performance penalty hit.

The user can still adjust the parameters used for defining the swapfile in Chicago, however the need to do this is reduced by intelligent use of system defaults. Figure 32 shows the new simplified swapfile configuration options, allowing the user to specify the minimum and maximum swapfile size to use.



☑ Reserve Disk Space as Extra Memory
◆ Default
◇ Custom
Hard Disk: C:\ 47MB Free
Set Minimum: 0
Set Maximum: No maximum

Figure 32. Virtual Memory Settings in Chicago are Simplified Over Windows 3.1

The Registry—Centralized Configuration Store

Chicago uses a mechanism called the *Registry* that serves as the central configuration store for user, application, and computer-specific information. The Registry solves problems associated with .INI files as used in Windows 3.1, and is a hierarchical database that stores system-wide information in a single location, making it easy to manage and support.

Problems with Windows 3.1 .INI Files

Windows 3.1 uses initialization (.INI) files to store system-specific or application-specific information on the state or configuration of the system. For example, the WIN.INI file is used to store state information about the appearance or customization of the Windows environment, the SYSTEM.INI file is used to store system-specific information on the hardware and device driver configuration of the system, and various .INI files are used to store application-specific information about the default state of an application (for example, WINFILE.INI, MSMAIL.INI, CLOCK.INI, CONTROL.INI, PROGMAN.INI, and so on).

Problems with .INI files under Windows 3.1 for configuration management include:

- Information is stored in several different locations including CONFIG.SYS, AUTOEXEC.BAT, WIN.INI, SYSTEM.INI, PROTOCOL.INI, private .INI files, and private .GRP files

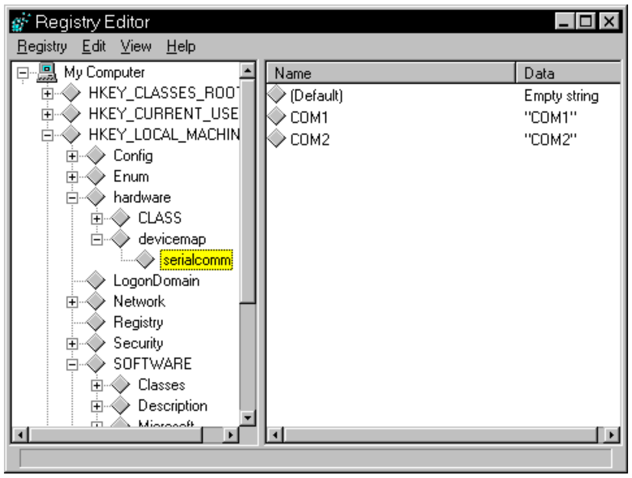
- .INI files are text-based, are limited to 64K in total size, and APIs only allow for get/write operations
- information stored in .INI files is non-hierarchical and supports only two-levels of information (i.e., key names broken up by section heading)
- Many .INI files contain a myriad of switches and entries that are complicated to configure or are used only by operating system components
- .INI files provide no mechanism for storing user-specific information, thus making it difficult for multiple users to share a single computer
- Configuration information in .INI files is local to each system, and no API mechanisms are available for remotely managing configuration, thus making it difficult to manage multiple systems

Solution to Windows 3.1 .INI File Problems

To solve problems associated with .INI files under Windows 3.1, the Registry was designed with the following goals in mind:

- Simplify the support burden
- Centralize configuration information
- Provide a means to store user, application, and computer-specific information
- Provide local and remote access to configuration information

The Registry is structured as a hierarchical database of keys, where each key can contain a value, or can even contain other keys (subkeys). While similar in some ways to the Registration Database used in Windows 3.1, which served as a central repository for file associations and OLE registration information, the Registry in Chicago extends the previous structure to support keys that can have more than one value and can also support data of different types. The Registry uses a hierarchical structure to store text or binary value information to maintain all of the configuration parameters normally stored in the Windows system .INI files such as WIN.INI, SYSTEM.INI, and PROTOCOL.INI.



Name	Data
(Default)	Empty string
COM1	"COM1"
COM2	"COM2"

Figure 33. Hierarchy of Registry as Displayed by the Registry Editor

The Registry is made up of several .DAT files that contain system-specific information (SYSTEM.DAT) or user-specific information (USER.DAT). System-specific information such as the static reference to loading virtual device drivers will be moved as appropriate from the SYSTEM.INI file to the Registry.

System Switch Simplification

Another improvement made over Windows 3.1 and its use of ..INI files is related to system switch simplification. Windows 3.1 supports over several hundred different configuration switches that can be specified in system .INI files including the WIN.INI or SYSTEM.INI files. With intelligent enhancements made to the system, and better dynamic configuration properties, Chicago has reduced the number of entries that are normally associated with .INI files. These reductions didn't come just by moving .INI entries to the Registry, but by examining and justifying the presence of each and every one.

.INI Files Still Exist for Compatibility Reasons

For compatibility reasons, WIN.INI and SYSTEM.INI and application-specific .INI files (as well as CONFIG.SYS and AUTOEXEC.BAT) do not go away. The Win16 APIs for manipulating .INI files will still manipulate .INI files, however Win32–based applications will be encouraged to use the Registry APIs to consolidate application-specific information.

Many existing Win16–based applications expect to find and manipulate the WIN.INI and SYSTEM.INI files to add entries or load unique device drivers, therefore SYSTEM.INI, for example, will still be examined during the Chicago boot process to check for virtual device drivers in the **[386Enh]** section.

Role in Plug and Play

One of the primary roles of the Registry in Chicago is to serve as a central repository for hardware-specific information for use by the Plug and Play system components. Chicago maintains information about hardware components and devices that have been identified through an enumeration process in the hierarchical structure of the Registry. When new devices are installed, the system checks the existing configuration in the Registry to determine the hardware resources (for example, IRQs, I/O addresses, DMA channels, and so on) that are not being used, so the new device can be properly configured without conflicting with a device already installed in the system.

Remote Access to Registry Information

Another advantage of the Registry for Win32–based applications is that many of the Win32 Registry APIs are remoted using the remote procedure call (RPC) mechanism in Chicago to provide access to Registry information across a network. This allows desktop management applications to be written to aid in the management and support of Windows–based computers, and allows the contents of the Registry on a given PC to be queried and over a network. With this mechanism, industry management mechanisms such as SNMP or DMI can easily be integrated into Chicago, simplifying the management and support burden of an MIS organization. See the "Chicago Networking" section later in this guide for more information on manageability and remote administration.

Better Font Support

Font support in Chicago has been enhanced to provide better integration with the Chicago Shell user interface, optimized for the 32-bit environment, and provides capabilities such as font smoothing for fonts that has not been offered previously as part of a mainstream desktop operating system.

32-bit TrueType Rasterizer

The rasterizer component for rendering and generating TrueType fonts is enhanced in Chicago. The rasterizer is written as a 32-bit component, and delivers better fidelity from the mathematical representation to the generated bitmap, as well as better performance for rendering TrueType fonts.

In addition to performance enhancements, the new 32-bit rasterizer also provides support for generating complicated glyphs (for example, Han), and results in a faster initial boot time when lots of fonts are installed in the system than Windows 3.1.

Support for Smoother Fonts

In Windows 3.1, TrueType provided a major improvement over the quality of displayed fonts over raster-based fonts provided in Windows 3.0. TrueType helped to provide smooth looking fonts that looked good from small sizes, up to big sizes.

However, as the font gets larger, the ability to rasterize the font to appear smooth degrades.

Chicago provides smoother looking TrueType fonts on the screen by using a technique called *antialiasing*. Put simply, anti-aliasing is to TrueType, what TrueType is to raster fonts. Normally, characters are displayed on the screen using the same intensity level and may appear to be somewhat jagged when displayed in large font sizes. Font smoothing in Chicago is accomplished by using a technique that uses different intensity levels at edges and corners to produce a resulting image that appears much smoother than the unsmoothed image. Antialiasing support for fonts requires a 256 color display mode (or higher), to support different intensity levels resulting in a smooth font appearance. Font smoothing support in Chicago preserves your existing software investment by working with any TrueType font.

Robustness Improvements

Chicago improves on the robustness of Windows 3.1 to provide great support for running MS-DOS, Win16, and Win32–based applications, and provides a high level of system protection from errant applications.

Windows 3.1 provided a number of mechanisms to support a more robust and stable environment over Windows 3.0. These improvements included:

- **Better resource cleanup.** When a Windows or MS-DOS–based application crashed, users were able to continue running such that they could save their work.
- **Local reboot.** This allowed users to shut down an application that hung.
- **Parameter validation for API calls.** This allowed the system to catch many common application errors and fail the API call, rather than allowing bad data to be passed to an API.

While the work done in Windows 3.1 provided a more robust and stable environment than Windows 3.0, we made it even better in Chicago.

System-wide Robustness Improvements

System-wide improvements resulting in a more robust operating system environment than Windows 3.1 include:

- Better local reboot
- Virtual device driver (VxD) thread cleanup when a process ends
- Per-thread state tracking
- Virtual device driver parameter validation

Better Local Reboot

The ability for a user to end an application or a virtual machine (VM) that hangs is called a *local reboot*. With Windows 3.1, users were able to perform a local reboot for an application or VM that the system thought was hung by pressing the three-key Ctrl-Alt-Del combination. Users could pretty easily end errant VMs with the local reboot request, however requesting a local reboot for a Windows–based application often resulted in bringing the entire system down or not allowing the user to end the errant Windows–based process.

Chicago greatly improves upon the local reboot support by providing a means to end an MS-DOS–based application running in a VM, end a Win16–based application, or end a Win32–based application, in a manner without bringing down the entire system. The process of cleaning up the system after a local reboot is now more complete than for Windows 3.1. This process is described more fully later in this chapter.

When a user requests a local reboot, the Chicago system displays a dialog box identifying the different tasks that are running and the state that the system perceives each to be in. This level of detail affords the user much more flexibility and control over local reboot than with Windows 3.1.

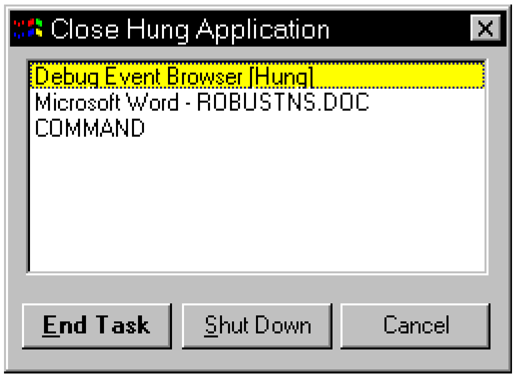


Figure 34. Local Reboot Dialog Box in Chicago

Applications are identified as "hung" when they haven't checked the message queue for a period of time. Although an application may be performing a computationally-intensive operation, a well-behaved application will check the message queue on a more frequent basis. Just as with Windows 3.1, it is necessary for a Win16–based application to check the message queue in order to relinquish control to other tasks running.

Virtual Device Driver Thread Clean-up When a Process Ends

Local reboot support is also aided by improved VxD thread clean-up when a given process ends. With Windows 3.1, it was quite common for the system to be unable to recover if the system was running real-mode code such as BIOS routines when an application ended abnormally, or if the user requested a local reboot to end a seemingly-hung application. For example, suppose the user requested a local reboot or suppose an operation (such as a network operation in real-mode, a disk I/O, or an asynchronous application request) ended abnormally because of another application-based error. In these cases, Windows 3.1 couldn't necessarily clean up properly to free allocated resources, and possibly couldn't even return control to the user.

Chicago improves system clean-up by providing each system VxD the ability to track the resources it allocates on a per-thread basis. Since most computer system functionality and support is handled by VxDs in Chicago rather than by real-mode code or BIOS routines, Chicago can recover from errors or situations that, under Windows 3.1, would required the computer to be rebooted.

When Chicago ends a given thread, each VxD receives notification that the thread is ended (because the user exited the application, a local reboot was requested, or the application ended abnormally). This notification allows the VxD to safely cancel any operations it is waiting to finish. This also frees any resources that the VxD previously allocated for the thread or application. Since the system tracks an entire VM, a Win16 application, and a Win32 thread, each as a per-thread instance, the system can clean up properly at each of these levels, without affecting the integrity of the system.

Per-Thread State Tracking

Resource tracking in Chicago is much better than that provided in Windows 3.1 to aid system clean-up. In addition to tracking resources on a per-thread basis by system VxDs, resources such as memory blocks, memory handles, graphics objects, and other system items are allocated and also tracked by system components on a per-thread basis. Tracking these resources on a per-thread basis allows the system to clean up safely when a given thread ends, either normally at the user's request, or abnormally. Resources are identified and tracked by both a thread ID, and by the major version number of Windows that is stored in the .EXE header of the application.

For a discussion of how the thread ID and the version number of Windows are used to facilitate cleanup of the system and recovery of allocated resources for Win16 and Win32–based applications, see the Win16 and Win32–based application robustness sections in this guide.

Virtual Device Driver Parameter Validation

Virtual device drivers are an integral part of the Chicago operating system and have a more important role than in Windows 3.1, as many operating system components are implemented as VxDs. To help provide for a more stable and reliable operating system, Chicago provides support for parameter validation of virtual device drivers, something that was not available for Windows 3.1. The debug version of Chicago system files provided as part of the Chicago SDK and Chicago DDK will aid VxD developers to debug their VxDs during the course of development to ensure their VxDs are stable and robust.

In addition to providing improved system-wide robustness, Chicago delivers improved robustness for running MS-DOS–based, Win16–based, and Win32–based applications, providing for a more stable and reliable environment than Windows 3.1.

Robustness for MS-DOS–based Applications

Chicago provides improved support for running MS-DOS–based applications under Chicago that were not possible with Windows 3.1. Several improvements present in

Chicago provide great robustness for running MS-DOS–based applications. These improvements are described in the next two sections.

Improved Protection for Virtual Machines

Each MS-DOS–based application runs in a separate VM, and are configured by default to execute preemptively and run in the background when another application is active. Each VM is protected from other tasks running in the system, and an errant Win16– or Win32–based application can't crash a running MS-DOS–based application, and vice versa.

Under Windows 3.1, each VM inherits the attributes and environment configuration from the global System VM. While each VM is protected from another VM preventing errant MS-DOS–based applications from accessing memory or overwriting system code thus possibly bringing the system down, the VM does not provide complete protection preventing an MS-DOS–based application from overwriting MS-DOS system code. MS-DOS–based applications have full access to all memory locations in the first megabyte of addressable memory space (i.e., the real-mode memory range).

Chicago supports a higher level of memory protection for running MS-DOS–based applications, preventing the applications from overwriting the MS-DOS system area in real-mode. Users can configure their MS-DOS–based applications to run with "general memory protection" enabled if they want the highest level of system protection. This mode is not enabled by default due to overhead required to validate memory access requests. Furthermore, parameter validation of Int 21h operations on pointers will be performed. This will increase the robustness of the system.

Better Cleanup When a Virtual Machine Ends

When a VM ends in Chicago—either normally because the exited the application or VM or requested a local reboot, or abnormally because the application ends abnormally—the system frees all resources allocated for the VM. In addition to the resources allocated and maintained by the system VxDs as previously discussed, the system tracks resources allocated for the VM by the Virtual Machine Manager, including DPMI and XMS memory that the VM requested.

In Windows 3.1, resources such as DPMI memory are not released properly when the VM is ended. Chicago frees the DPMI memory used by the VM and other resources allocated by the operating system components.

Robustness for Win16–based Applications

Chicago provides improved support for running Win16–based applications. It also provides great robust Win16 application support plus compatibility with existing Windows–based applications, while keeping the memory requirements low. The next two sections describe improvements for Win16–based applications running under Chicago.

Per-Thread State Tracking

Under Windows 3.1, when a Windows–based application ended, the resources used by the application were not released by the system. Some Windows–based applications took this into account and didn't free certain resources as the allocated resources could then be accessed by other in-memory Windows–based applications or system components (such as DLLs). Changing the way the system behaves when a Win16 application ended—for example, by freeing up all resources allocated to the Win16 application immediately—might break an existing application.

Under Chicago, each Win16–based application runs as a separate thread in the Win16 address space to facilitate resource tracking. When a Win16 application ends, resources allocated to the Win16 application aren't immediately released by the system but are held by the system until the system can safely free them. When the last Windows 3.*x* application is ended, Chicago determines that it is safe to free all resources allocated for Win16–based applications and begins cleaning the system of resources associated with Windows 3.*x* applications. Chicago determines that no more Win16–based applications are running by associating the Windows version number of the application with the thread ID for the running process. When no more Windows 3.x applications are running in the system, Chicago frees any remaining resources allocated by the Win16–based applications.

Parameter Validation for Win16 APIs

Chicago provides support for checking the validity of parameters passed to Windows APIs by Win16–based applications. Some users perceived Windows 3.0 to be unstable because the "Unrecoverable Application Errors" (UAE) were common when working with Windows–based applications. Most of this instability was in fact caused by Windows–based applications that passed invalid parameters to Windows API functions. The APIs in turn attempted to process this bad data and usually attempted to access an invalid area of memory. For example, when an application that inadvertently passed a NULL pointer to a Windows API function which tried to access memory at the address referenced, it would generate a UAE or "general protection fault."

Chicago provides parameter validations for all Win16–based APIs and checks incoming data to API functions to ensure the data is valid. For example, functions that reference memory are checked for NULL pointers, and functions that operate on data within a range of values are checked to ensure the data is within the proper range. If invalid data is found, an appropriate error number will be returned to the application. It is then up to the application to catch the error condition and handle it accordingly.

The Chicago SDK provides debug system components to aid software developers to debug their applications. The Chicago debug components provide extensive error reporting for parameter validation to aid the developer in tracking common problems related to invalid parameters during the course of development.

Robustness for Win32–based Applications

While the robustness improvements for running MS-DOS–based and Win16–based applications in Chicago is better than that provided by Windows 3.1, the greatest support for robustness in Chicago is available when running Win32–based applications. Win32–based applications also benefit from preemptive multitasking, linear address space (rather than segmented), and support for a feature-rich API set.

Robustness support for Win32–based applications includes:

- A private address space for each Win32–based application to run, segregating and protecting one application from others that are running concurrently
- Win32 APIs that support parameter validation to provide for a stable and reliable environment
- Resources are tracked by threads and freed up immediately when the thread ends
- Separate message queues are used to ensure that a hung Win32–based application will not suspend the entire system

Each Win32–based Application Runs in its own Private Address Space

Each Win32–based application runs in its own private address space. This provides protection of its resources at the system level from other applications running in the system. It also prevents other applications inadvertently overwriting the memory area of a given Win32–based application, and prevents the Win32–based application from inadvertently overwriting the memory area of another application or the system as a whole.

Parameter Validation for Win32 APIs

As with parameter validation for Win16–based applications, Chicago provides parameter validation for Win32 APIs used by Win32–based applications. The Chicago SDK helps software developers debug errors resulting from attempts to pass invalid parameters to Windows APIs. For additional information about parameter validation for Win16 APIs, see the discussion of robustness for Win16–based applications presented earlier in this guide.

Per-Thread Resource Tracking

Resources allocated by threads in Win32–based applications are tracked by the system. Unlike thread tracking for Win16–based applications, Chicago resources are automatically deallocated when the thread ends processing. This helps to ensure that allocation system resources are freed immediately and are available for use by other running tasks.

Resources are cleaned up properly when threads either end execution on their own (for example, perhaps the developer inadvertently did not free allocated resources), or when the user requests a local reboot that ends a given Win32 application thread or process. Unlike Win16–based applications designed to run under Windows 3.1, Win32–based applications free up allocated resources immediately when the application or a separate thread ends.

Separate Message Queues for Win32–based Applications

The Windows environment performs tasks based on the receipt of messages sent by system components. Each message is generated based on an action or *event* that occurs on the system. For example, when a user presses a key on the keyboard and releases it, or moves the mouse, a message is generated by the system and passed to the active application informing it of the event that occurred. Windows–based applications call specific Windows API functions to extract event messages from message queues and perform operations on the messages (for example, accept an incoming character typed on the keyboard, or move the mouse cursor to another place on the screen).

Under Windows 3.1, a single message queue was used by the entire system. Win16–based applications cooperatively examined the queue and extracted messages destined to them. This single-queue scheme posed some problems. For example, if a

Win16–based application hung and prevented other applications from checking the message queue, the message queue became full and accepted no new messages. Then other Win16–based applications were suspended until control was relinquished to them and they were able to check for event messages.

Chicago solves the problems inherent with a single message queue in Windows 3.1, by providing for separate message queues for each running Win32–based application (see Figure 35). The system takes messages from the input message queue and passes them to the message to the appropriate Win32–based application or to the Win16 Subsystem, if the message is destined for a Win16–based application. If a Win32–based application hangs and no longer accepts and processes incoming messages destined for it, the Win32–based application does not affect other Win16– and Win32–based applications currently running.

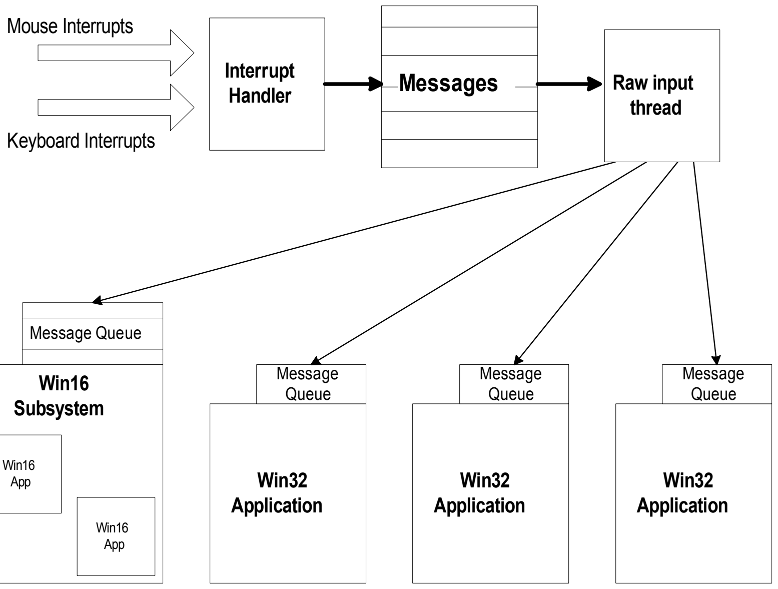


Figure 35. Win32–based Applications Use Separate Message Queues for Increased Robustness

If a Win32–based application ends or the user requests a local reboot operation on a Win32–based application, having separate message queues improves the robustness of the operating system by making it easier to clean up and to free system resources used by the application. It also provides greater reliability and recoverability if an application hangs.

Improved Local Reboot Effectiveness

Due to the robustness improvements supported in the system for Win32–based applications (including the use of a private address space, separate message queues, and resource tracking by thread), users should be able to end ill-behaved Win32–

based applications in almost all cases, without affecting the integrity of the Windows system or other running applications.

When a Win32–based application is ended, resources are deallocated and cleaned up by the system as soon as the application ends. Because Win32–based applications run in their individually-allocated environment, this method is even more robust than the way Chicago is able to reallocate Win16 application resources. (See the section called “Robustness for Win16-based Applications” for more details.)

Structured Exception Handling

An *exception* is an event that occurs during the execution of a program, and that requires the execution of software outside the normal flow of control. Hardware exceptions can result from the execution if certain instruction sequences, such as division by zero or an attempt to access an invalid memory address. A software routine can also initiate an exception explicitly.

The Microsoft Win32 application programming interface (API) supports *structured exception handling*, a mechanism for handling hardware- and software-generated exceptions. Structured exception handling gives programmers complete control over the handling of exceptions. The Win32 API also supports termination handling, which enables programmers to ensure that whenever a guarded body of code is executed, a specific block of termination code is also executed. The termination code is execute regardless of how the flow of control leaves the guarded body. For example, a termination handler can guarantee that clean-up tasks are performed even if an exception or some other error occurs while the guarded body of code is being executed. Structured exception and termination handling is an integral part of the Win32 system and it enables a very robust implementation of system software.

Chicago provides structured exception and termination handling for Win32–based applications that make use of this functionality—resulting in applications that can identify and rectify error conditions that may occur outside their realm of control, providing a more robust computing environment.

Try It!

Mouse

To see how the robustness improvements made in Chicago results in a more stable, and reliable environment than Windows 3.1, you’ve got to try it!

Local Reboot

To see how local reboot works in Chicago and Windows 3.1, you’ve got to try the three-finger salute. With a couple of applications running in the system, press CTRL-ALT-DEL simultaneously.

Under Windows 3.1, the system may identify the currently active application as the application that has the focus of the local reboot request, or may report back that there is no application in a hung or inactive state.

Under Chicago, the user will be presented with a list of active applications, and is given the option of terminating any currently running tasks. Applications that are no longer responding to the system are identified as being “hung” in the local reboot dialog box.

Improved Support for Running MS-DOS–based Applications

Support for MS-DOS–based applications, device drivers, and terminate-and-stay-resident programs (TSRs) does not go away in Chicago. In fact, Chicago offers better compatibility for running MS-DOS–based applications than Windows 3.1 does, including applications that are hardware-intensive, such as games.

As with Windows 3.1, Chicago provides the ability for a user to launch an MS-DOS command prompt as an MS-DOS virtual machine (VM). The functionality supported in an MS-DOS VM, is the same functionality that is available under the latest version of MS-DOS, allowing users to run the same intrinsic commands and utilities.

Chicago delivers great support for running MS-DOS–based applications, allowing even applications that would not run under Windows 3.1 to run properly. This allows MS-DOS–based applications to coexist peacefully with the rest of the Chicago environment.

Summary of Improvements over Windows 3.1

Improvements made in the system provide the following benefits for running MS-DOS–based applications in the Chicago environment:

- Zero conventional memory footprint for protected-mode components
- Improved compatibility for running MS-DOS-based applications
- Improved robustness for MS-DOS–based applications
- Better support for running MS-DOS–based games, including in a window
- Support for running existing MS-DOS–based applications without exiting Chicago or running MS-DOS externally
- Consolidated attributes for customizing properties of MS-DOS–based applications
- Toolbar availability when running an MS-DOS–based application in a window providing quick access to features and functionality to manipulate the window environment
- User-scaleable MS-DOS window through the use of TrueType fonts
- Ability to gracefully end MS-DOS–based application without exiting the application
- Ability to specify local VM environment settings on a per-application basis through the use of a separate batch file

- Support for new MS-DOS commands providing tighter integration between the MS-DOS command line and the Windows environment

Zero Conventional Footprint Components

Chicago helps to provide the maximum amount of conventional memory available for running existing MS-DOS–based applications. Some MS-DOS–based applications do not run under Windows 3.1 because by the time MS-DOS–based device drivers, MS-DOS–based TSRs, MS-DOS–based networking components, and Windows 3.1 were loaded, there was not enough conventional memory available. Chicago provides 32-bit protected-mode components that replace many of the 16-bit real-mode counterparts, providing the same functionality while improving overall system performance and using no conventional memory.

32-Bit virtual device drivers are provided to replace the 16-bit real-mode counterparts for such functions as:

Description	**File(s)**	**Conventional Memory Saved**
Microsoft Network client software	NET.EXE (full)	95K
	PROTMAN	3K
	NETBEUI	35K
	EXP16.DOS (MAC)	8K
Novell NetWare client software	LSL	5K
	EXP16ODI (MLID)	9K
	IPXODI.COM	16K
	NETBIOS.EXE	30K
	NETX.EXE	48K
	VLM.EXE	47K
MS-DOS extended file sharing and locking support	SHARE.EXE	17K
Adaptec SCSI driver	ASPI4DOS.SYS	5K
Adaptec CD-ROM driver	ASPICD.SYS	11K
Microsoft CD-ROM Extensions	MSCDEX.EXE	39K
SmartDrive disk caching software	SMARTDRV.EXE	28K
Microsoft Mouse driver	MOUSE.COM	17K

The resulting memory savings for using 32-bit protected-mode components can be quite dramatic. For example, suppose a PC was configured with the NetWare 3.*x* client software, using a SCSI CD-ROM drive, and with the MS-DOS support files SMARTDrive and Mouse. The resulting conventional memory savings that Chicago would offer this configuration would be *over* 225 KB!

Improved Compatibility

There are a number of reasons why some MS-DOS–based applications do not run properly under Windows 3.1. For example, some MS-DOS applications required lots of available free conventional memory, and thus wouldn't run in a DOS VM due to large real-mode components, such as network drivers or device drivers. Other MS-DOS–based applications would not run under Windows 3.1 because they required direct access to the computer hardware and conflicted with Windows internals or other device drivers.

The goal of Chicago to support running MS-DOS–based applications is to be able to run existing "clean" MS-DOS–based applications that ran under Windows 3.1, as well as to support running the "bad" MS-DOS–based applications that tried to take over the hardware or required machine resources unavailable under Windows 3.1.

Many MS-DOS–based games take advantage of the assumption that they are the only application running in the system, and access and manipulate the underlying hardware directly, thus preventing them from being run in a MS-DOS VM under Windows 3.1. Games are the most notorious class of MS-DOS–based applications that don't get along well with Windows 3.1. Some of these applications write to video memory directly, manipulate the hardware support resources such as clock timers, and take over hardware resources such as sound cards.

A number of things have been done to provide better support for running MS-DOS–based applications that interact with the hardware, including better virtualization of computer resources such as timers and sound device support. In addition, the use of 32-bit protected-mode device drivers benefits MS-DOS–based applications by providing them with more free conventional memory than was available under Windows 3.1, allowing a class of memory-intensive applications to run properly.

Different MS-DOS–based applications require varying levels of support from both the computer hardware and from the operating system. For example, there are some MS-DOS–based games that require close to 100% use of the CPU to perform properly, and there are other MS-DOS–based applications that modify interrupt addresses and other low-level hardware settings. Chicago provides several different levels of support for running MS-DOS–based applications. These levels of support take into account that different applications interact with the hardware in different ways—some behave well, whereas others expect exclusive access to the PC system and hardware. By default, MS-DOS–based applications are preemptively multitasked with other tasks running on the system and can run either full-screen or in a window. (CPU-intensive MS-DOS–based applications may not run well in a window for performance reasons, but can be run in full-screen mode to get the best response level.)

Single MS-DOS Application Mode

To provide support for the most intrusive set of MS-DOS–based applications that only work under MS-DOS and require 100% access to the system components and system resources, Chicago provides a mechanism that is the equivalent of running an MS-DOS–based application from real-mode MS-DOS—this mechanism is called Single MS-DOS application mode. While fewer MS-DOS–based applications will need to run in this mode due to improved compatibility support provided by Chicago, this mode provides an "escape hatch" mechanism for running applications that only run under MS-DOS.

To run an MS-DOS–based application in this mode, users set the Single MS-DOS Application Mode property from the Program tab on the MS-DOS property sheet for the application. In this mode, Chicago removes itself from memory (except for a small stub), and provides the MS-DOS–based application with full access to all the resources in the computer. Before a user runs an MS-DOS–based application in this mode, Chicago prompts the user as to whether running tasks can be ended. Upon user's approval, Chicago ends all running tasks, load a real-mode copy of MS-DOS, and launch the specified application. This process is like exiting Windows 3.1, then running the specified MS-DOS–based application under MS-DOS. Once the user exits the MS-DOS–based application, Chicago restarts and returns the user to the Chicago shell.

Improved Support for Graphic-intensive MS-DOS–based Applications

Chicago improves the support for running MS-DOS–based applications in the Windows environment by providing better support for running graphic-based applications in a window, rather than requiring the application to be run in full-screen mode as with Windows 3.1. MS-DOS–based applications that use VGA graphic video modes can now be run in an MS-DOS window, whereas under Windows 3.1 the user was prevented from doing this. While Chicago is improved over Windows 3.1, the user may choose to run graphic-intensive MS-DOS–based applications in full-screen mode for the best level of performance.

Improved Memory Protection

To support a higher level of memory protection for running MS-DOS–based applications, Chicago includes a "global memory protection" attribute on the Program property sheet tab that allows the MS-DOS system area to be protected from errant MS-DOS–based applications. When the global memory protection attribute is set, the MS-DOS system area sections are read-protected so that applications can't write into this memory area and corrupt MS-DOS support and MS-DOS–based device drivers. In addition to the system area protection, enhanced parameter validation is performed for file I/O requests issued through the MS-DOS INT 21h function, providing a higher degree of safety.

This option is not enabled by default for all MS-DOS–based applications due to the additional overhead associated with providing improved parameter and memory address checking. Users would set this flag if they are constantly encountering difficulty running a specific MS-DOS–based application.

Better Defaults for Running MS-DOS–based Applications

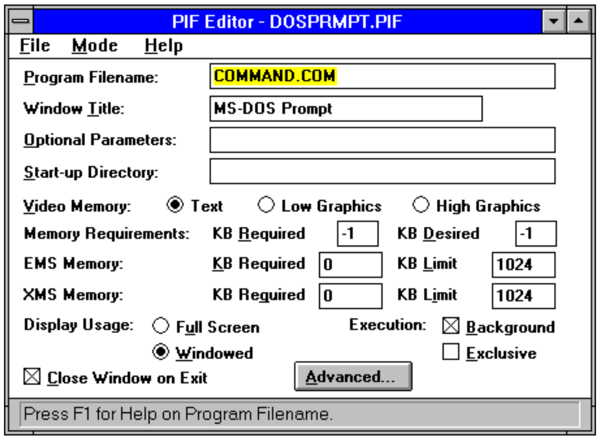
By default, Windows 3.1 runs MS-DOS–based applications full-screen and disabled the ability for the MS-DOS–based application to run in the background. To change this default behavior, it was necessary for users to use the PIFEDIT application and modify or create a program information file (.PIF) for the given MS-DOS–based application.

Chicago defaults to running MS-DOS–based applications in a window, and enables the background execution setting, allowing the application to continue to run when it is not the active application. The change in this default behavior provides better integration between running MS-DOS–based applications and Windows–based applications without requiring the user to change or customize the state of the system.

Consolidated Customization of MS-DOS–based Application Properties

Each MS-DOS–based application has different characteristics and mechanisms for using machine resources such as memory, video, and keyboard access. Chicago (and Windows 3.1) understand how to run Windows–based applications as requests for system services is handled through the use of the Windows API. However, MS-DOS–based applications only included minimal information about their requirements in the format of the .EXE header associated with each application. To provide additional information to the Windows environment about the requirements for running MS-DOS–based applications, a program information file (.PIF) is used to specify the configuration settings used to run MS-DOS–based applications in the Windows environment.

Under Windows 3.1, the PIF Editor application was used to create or change properties associated with running MS-DOS–based applications. Problems associated with the PIF Editor or PIF creation process included difficulty in accessing the PIF editor or PIF settings, the disassociation of PIF properties from the MS-DOS–based application for new users, the lack of a single location for storing PIF files beyond placing them all in the WINDOWS directory, and less-than-intelligent defaults for running MS-DOS–based applications.


PIF Editor - DOSPRMPT.PIF

File Mode Help

Program Filename: COMMAND.COM

Window Title: MS-DOS Prompt

Optional Parameters:

Start-up Directory:

Video Memory: [✓ Text] Low Graphics High Graphics

Memory Requirements: KB Required -1 KB Desired -1

EMS Memory: KB Required 0 KB Limit 1024

XMS Memory: KB Required 0 KB Limit 1024

Display Usage: Full Screen [✓ Windowed]

Execution: [✓ Background] Exclusive

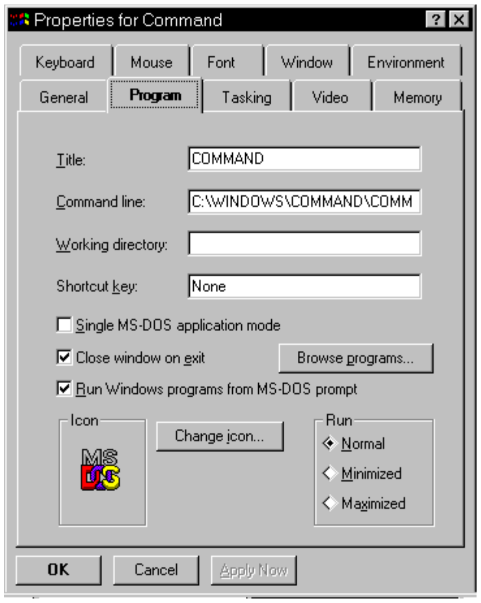
[✓ Close Window on Exit]

Advanced...

Press F1 for Help on Program Filename.

Figure 36. PIF Editor in Windows 3.1

Chicago enhances the ability to define properties for running MS-DOS–based applications by consolidating PIF files into a single location (the PIF directory where Chicago is installed), providing easy access to property information for an application (using the secondary mouse button to simply click the icon or application window), and simplifying the user interface to provide better organization of property settings (through the use of a tabbed property sheet dialog box). Through the use of property sheets, Chicago provides greater flexibility and control for running MS-DOS–based applications.


Properties for Command

Keyboard | Mouse | Font | Window | Environment

General | Program | Tasking | Video | Memory

Title: COMMAND

Command line: C:\WINDOWS\COMMAND\COMM

Working directory:

Shortcut key: None

Single MS-DOS application mode

[✓ Close window on exit]

Browse programs...

[✓ Run Windows programs from MS-DOS prompt]

Icon

Change icon...

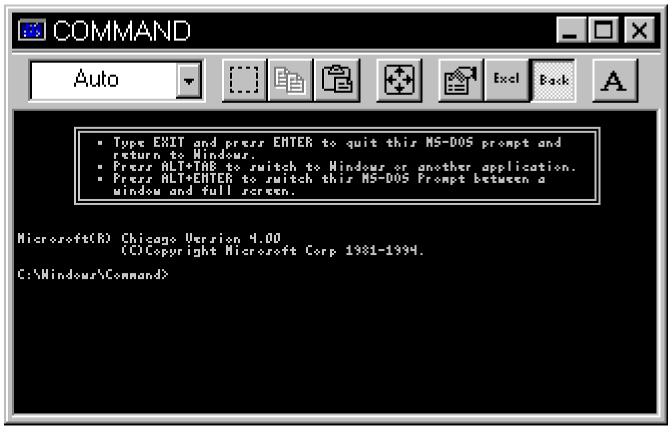
Run: [✓ Normal] Minimized Maximized

OK Cancel Apply Now

Figure 37. Property Sheet for Configuring an MS-DOS–Based Application

Toolbar in MS-DOS Window

In addition to providing compatibility enhancements in Chicago to support running MS-DOS–based applications better than under Windows 3.1, Chicago makes it even easier to use MS-DOS–based applications in the Windows environment than Windows 3.1. Many Windows–based applications implement a toolbar to provide quick access to common features and functionality of a product, Chicago extends this simplicity and power feature to making it easy to access functionality associated with an MS-DOS–based application.



```
  • Type EXIT and press ENTER to quit this MS-DOS prompt and
    return to Windows.
  • Press ALT+TAB to switch to Windows or another application.
  • Press ALT+ENTER to switch this MS-DOS Prompt between a
    window and full screen.

Microsoft(R) Chicago Version 4.00
             (C)Copyright Microsoft Corp 1981-1994.

C:\Windows\Command>
```

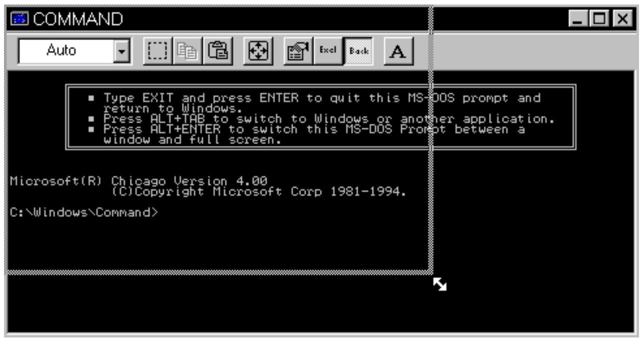
Figure 38. Toolbar in Windowed MS-DOS Box

Optionally, a user can enable the display of a toolbar in the window of a running MS-DOS–based application to provide the user with quick access to the following functionality:

- Simpler access to cut, copy, and paste operations for integrating text-based or graphics-based MS-DOS–based applications with Windows–based applications
- Easy access to switching from windowed to full-screen mode
- Quick access to property sheet information associated with the MS-DOS–based application
- Access to MS-DOS VM tasking properties such as exclusive or foreground processing attributes
- Easier access to font options for use in displaying text in a windowed MS-DOS VM

User-Scalable MS-DOS Window

Chicago supports the use of a TrueType font in a windowed MS-DOS VM, supporting the ability for a user to scale the MS-DOS window to any size. When the font size is set to “Auto,” the MS-DOS window is sized automatically to display the entire window within the user-specified area. The figure below shows the MS-DOS command prompt window being changed to a smaller size.

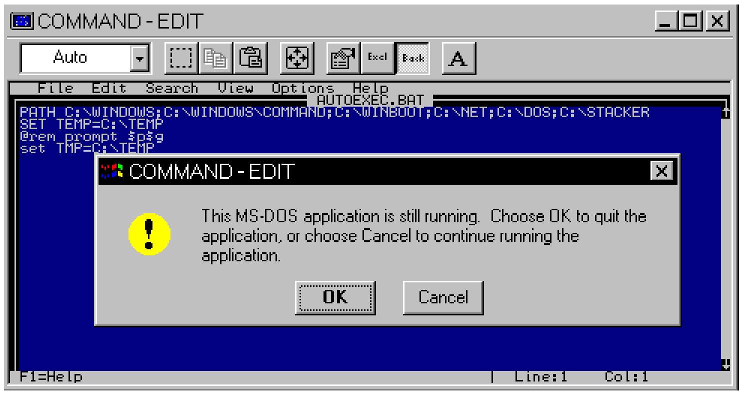


```
■ Type EXIT and press ENTER to quit this MS-DOS prompt and
  return to Windows.
■ Press ALT+TAB to switch to Windows or another application.
■ Press ALT+ENTER to switch this MS-DOS Prompt between a
  window and full screen.

Microsoft(R) Chicago Version 4.00
             (C)Copyright Microsoft Corp 1981-1994.

C:\Windows\Command>
```

Figure 39. With TrueType Font Support, Users Can Scale an MS-DOS Window

Ending MS-DOS–based Applications Graceful

Chicago provides support for gracefully closing an MS-DOS VM through a property sheet setting available on an application-by-application basis. When enabled, the user can close an MS-DOS–based application just as a Windows–based application is closed—by clicking the close window button. If Chicago senses that the MS-DOS VM window contains a running MS-DOS–based application, Chicago prompts the user for confirmation to end the given application.



```
PATH C:\WINDOWS;C:\WINDOWS\COMMAND;C:\WINBOOT;C:\NET;C:\DOS;C:\STACKER
SET TEMP=C:\TEMP
@rem prompt $p$g
set TMP=C:\TEMP
```

This MS-DOS application is still running. Choose OK to quit the application, or choose Cancel to continue running the application.

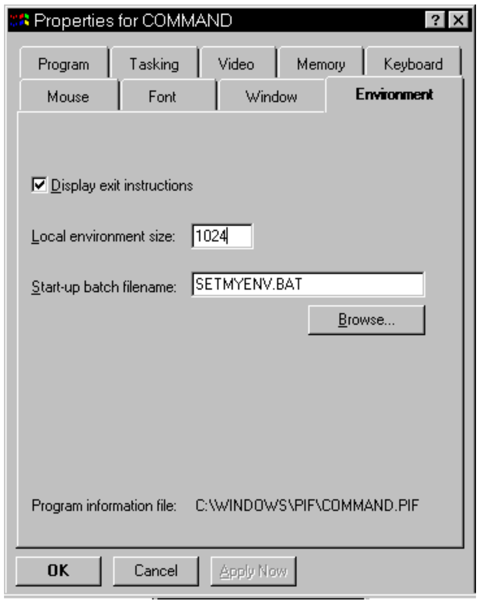
Figure 40. Warning Dialog Box Displayed When an MS-DOS—based Application is Active

In addition to simply ending an MS-DOS–based application, robustness improvements made to the Chicago system ensure that system clean up is completed properly and all allocated resources are freed. This results in memory used by the MS-DOS–based applications is deallocated properly and available for use by other applications. (Windows 3.1 didn't properly free DPMI memory, for example.)

Local Virtual Machine Environment Settings

When Windows 3.1 starts up, it uses the MS-DOS environment as specified before Windows is started as the default state for each MS-DOS VM that is created. Any TSRs or other memory resident software that is loaded before starting Windows is replicated across all MS-DOS VMs, whether the VM needs it or not. Windows 3.1 did not provide a mechanism to allow a user to run a batch file that set the VM environment, before starting a given MS-DOS–based application. Actually, a batch file could be run by the user under Windows 3.1, but once the batch file finished processing the command statements, the MS-DOS VM was closed.

Under Chicago, a batch file can be optionally specified for a given MS-DOS–based application allowing customization of the VM on a local basis before running the MS-DOS–based application. This allows MS-DOS environment variables to be set or customized for individual MS-DOS–based applications, and for TSRs to be loaded in the local VM only. This is like having a separate AUTOEXEC.BAT for different MS-DOS–based applications. The batch file is specified on the Environment tab of the property sheet for the MS-DOS–based application.



☑ Display exit instructions

Local environment size: 1024

Start-up batch filename: SETMYENV.BAT

Program information file: C:\WINDOWS\PIF\COMMAND.PIF

Figure 41. Property Sheet Tab for Specifying Environment Attributes

Support for UNC Pathnames to Access Network Resources

Chicago makes it even easier to access network resources from the MS-DOS command prompt by supporting the use of universal naming conventions (UNC). UNC names provide a standard naming scheme to reference network servers, and shared directories and use the following syntax: \\servername\sharename[\pathname]

The Chicago shell allows users to browse and connect to network servers without mapping a drive letter to the network resource. Chicago supports the same functionality at an MS-DOS command prompt and allows the user to:

- View the contents of shared directories on network servers from both Microsoft Network servers and Novell NetWare servers: **dir** \\servername\sharename[\pathname]
- Copy files from the contents of shared directories on network servers from both Microsoft Network servers and Novell NetWare servers: **copy** \\servername\sharename\pathname\file destination
- Run applications from shared directories on network servers for both Microsoft Network servers and Novell NetWare servers: \\servername\sharename\pathname\filename

New MS-DOS Prompt Commands

The MS-DOS command processor and utilities have been enhanced to provide better integration between MS-DOS functionality and the Windows environment. Commands that manipulate files have been extended to support long file names, and some new commands have been added to Chicago, providing access to new capabilities supported by the system.

Starting MS-DOS *and* Windows–based Applications

For example, the **start** command allows a user to start a Windows–based or MS-DOS–based application from the command prompt in one of the following ways:

start <*application name*> | <*document name*>

- Start an application by specifying the name of a document to open, and Chicago will launch the application associated with the given file type. For example, a user can type "**start myfile.xls**" and the application associated with the file specification will start, if there is a valid association.
- Start an MS-DOS–based application in a different MS-DOS VM instead of the current one.

- Start a Windows–based application from an MS-DOS command prompt. When the user just types the name of a Windows–based application it is essentially the same as typing "**start** <*application*>".

Support for Long File Names

Many MS-DOS intrinsic commands and utilities have been extended to support the use of long file names. Basic examples of support for long file names includes the following commands. Many other commands have also been extended.

- The **dir** command has been extended to show long file names in the directory structure, along with the corresponding 8.3 filename. Also, the **dir** command now supports a "verbose" mode to display additional file details by typing "**dir /v**".

- The **copy** command has been extended to allow copying or long file names to/from short or long file names. For example, typing: "**copy myfile.txt "this is my file"**" will create a new file with a long file name.

Try It!

Mouse

To see how Chicago improves support for running MS-DOS–based applications over Windows 3.1, you've got to try it!

Improved Support for Running MS-DOS–based Applications

- Try an MS-DOS–based application that is known to not run under Windows 3.1 and run it under Chicago. Does it work? (If not, report it as a bug)

- Take an MS-DOS–based application that is known to run under Windows 3.1, but doesn't run in a window, and run it under Chicago in a window. Does it work? (If not, report it as a bug)

More Free Conventional Memory

Install Chicago on a PC with a configuration similar to one now running Windows 3.1 with MS-DOS–based device drivers and TSRs loaded. For example, use PCs with SCSI drivers, network drivers, or system support files such as SMARTDRV, MSCDEX, or SHARE.

- Type the "**mem /c**" command under Windows 3.1 and under Chicago. Is there a memory savings under Chicago for the same configuration?

MS-DOS–based Application Property Sheets

To see a property sheet for an MS-DOS–based application, try the following:

- Use the secondary mouse button to simply click the icon for an MS-DOS–based application, and select the Properties... item.

- Use the secondary mouse button to simply click the title bar of an active MS-DOS–based application, and select the Properties... item.

Scalable MS-DOS Window

To demonstrate the ability to scale an MS-DOS window, open an MS-DOS VM window and set the font size to “Auto” from the Font tab on the property sheet.

Click the mouse in the scale region of the lower-right corner of the window and change the size—this functionality is more noticeable when performed at higher resolutions.

Launching Applications from the MS-DOS Command Prompt

To demonstrate the ability for launching applications under Chicago from the MS-DOS command prompt, try the **start** command in a variety of scenarios. From an MS-DOS command prompt, try these operations:

- Type “**start /?**” to see the options available.
- Type “**start edit**” to start the MS-DOS Edit application in another VM.

Type “**start /m clock**” to start the Clock Windows-based application in minimized form.

Plug and Play

Configuring PC hardware and operating systems has become a significant problem in the PC industry, resulting in customer dissatisfaction and increased support costs—all of which impacts the industry affecting PC market growth. A broad-based group of PC industry members is tackling this industry-wide problem with the development of an open and extensible framework architecture called Plug and Play. The Plug and Play specifications describe hardware and software changes to the PC and its peripherals that free the PC user from manually configuring hardware resources.

Chicago is the operating system that ties Plug and Play components together. Operating system services are implemented in Chicago to make PCs even easier to use by providing:

- Help in device detection for installing and configuring devices
- Event notification for informing other system components and applications of dynamic changes to the system state
- Tight integration among device drivers, system components, and the user interface to make the operating system easier to use, configure, and manage.

Plug and Play in Chicago not only offers functionality to make it even easier to use a Plug and Play PC, but provides benefits to configuring and managing legacy PC hardware.

The Problem With PCs Today

For a user who is not a trained technician, installing or configuring a device on a PC can be a daunting task. Most users have neither the time nor the inclination to learn about such arcane subjects as interrupt request (IRQ) lines, direct memory access (DMA) channels, small computer system interface (SCSI) termination, or monitor timings. However, if users want to add devices to their PCs or take advantage of the features of a new device, they often must address these subjects, because most existing PC systems offer no alternative. Potential PC users hear about problems that current users encounter in these areas, which reinforce their viewpoint that PCs are complex, intimidating, and difficult to use.

Although the availability of add-on devices is an advantage of the PC, the fact that the typical PC contains devices made by numerous vendors tends to compound the hardware and software configuration problem. The hardware, operating system, and applications don’t know about other PC components, and the hardware can’t tell when conflicts exist between different devices trying to share the same system resource.

The main problems associated with today’s PC hardware and operating systems can be summarized by the following three points:

- **Adding devices to a PC can be a painful process.**

  A lack of coordination between hardware and software components leads to device conflicts when vying for valuable system resources such as IRQs, DMA addresses, and I/O addresses.

  There is also no easy access to information about the configuration of a PC, leading to confusion and an increased burden on the end-user and technical support resources to solve conflicts and other device errors.

- **Software has no idea what's in the system.**

  Today's operating systems only support rudimentary mechanisms for allowing applications to query the configuration of a PC. This information usually covers just basic properties of the PC including the type of CPU it has, the amount of memory configured, and possibly information about base devices such as communication ports. However, beyond basic properties, there are no consistent mechanisms to query detailed information about connected peripherals, or support for receiving system notification that may be associated with dynamic configuration of system resources (such as the addition or removal of a device on-the-fly).

- **Evolution of the PC platform is stalled due to compatibility problems.**

  Many different bus standards exist that are used in PCs today. These include ISA, EISA, Micro Channel, PCMCIA, serial ports, parallel ports, and ECP. Creating a new bus standard or device architecture, while maintaining compatibility with the existing architectures is a difficult task. Plug and Play provides a framework on which to design and implement new PC architectures, providing a common and consistent way for devices to interact and coexist, using a bus-independent design.

Mobile Computers Demand Much Higher Flexibility

The bigger problem that the current PC architecture encounters is trying to support the higher flexibility requirements for mobile computers. Mobile computers are used in a number of environments by on-the-road users, and the technology aimed at mobile computing professionals is growing by leaps and bounds. The configuration scenario shared by mobile computer users is different from that of desktop computer users. The mobile environment is much more dynamic and demands higher flexibility from the computing platform:

- **Mobile users need flexible configuration support whether in the office and on the road.**

  Users plug their mobile PCs into a docking station while used within the office, and run them in an undocked state while on the road. While connected to a docking station, a mobile PC may have network connectivity for accessing shared corporate resources, however once it is undocked, it's necessary to reconfigure the PC, perhaps support network connectivity through a dial-up process rather than a local, physical connection.

- **Support for hot-docking or hot-plugging of devices needs special operating system support and applications aware of changing environment.**

  The advent and popularity of PCMCIA also poses some dilemmas for the operating system and application programs. A main issue is how best to provide support for dynamic configuration when a device is added or removed from the system. For example, what should the operating system or application do in response to the addition of a PCMCIA card that provides access to SCSI devices, provides additional hard disk storage, or adds modem connectivity to the PC? Any of these hardware changes may affect the way software behaves on the system. Therefore, it is necessary for the operating system to support a mechanism for notification to inform the applications that their system configuration state may change and that they will need to take appropriate action. For example, suppose someone uses a word processing application to open a document on a PCMCIA hard disk drive, then decides he or she want to remove that hard disk. To gracefully handle this situation, the word processing application (which, of course, is Plug-and-Play aware) saves and closes the document before the hard disk is removed.

The Plug and Play Solution

Through automatic installation of drivers and seamless configuration, the Plug and Play architecture will turn the PC into more of an “appliance” rather than a complex, difficult to configure piece of hardware as it is today. A key benefit of Plug and Play is that it will help to create and support a dynamic platform by recognizing and enabling the transformation of the PC platform to a more mobile and dynamic environment.

The Plug and Play architecture is an open, flexible, and cost-effective framework for designing Plug and Play products. Plug and Play was jointly developed by a group of leading vendors who obtained reviews for their design proposals from hundreds of companies in the industry. Plug and Play provides a framework that works on many types of bus architectures — ISA, EISA, Micro Channel, PCMCIA, VESA local bus (VL-bus), Peripheral Component Interconnect local bus (PCI), and so on — and I/O port connections, and can be extended to future designs.

Here are three major benefits of the Plug and Play architecture:

- **Support costs are reduced for end-users, MIS support organizations, and industry hardware and software vendors.**

  Reducing the complexities of installing and configuring devices and peripherals will have a material benefit for both users, and MIS organizations.

As many as half of all support calls currently received by operating system and device manufacturers are related to installation and configuration of devices. For businesses, reducing the high cost of supporting PCs increases the use of PCs in the workplace and focuses information systems personnel on using computer technology to solve business problems. Both Plug and Play PCs *and* legacy PCs store hardware and software configuration in the Registry for centralized access, so support benefits can be achieved on existing hardware.

- **Plug and Play makes it easy to install and configure add-on devices with little or no user intervention.**

  Chicago stores all information about the hardware and resource configuration of peripheral devices (such as IRQs, I/O addresses, and memory addresses) in the Registry. On Plug and Play PCs, resource allocation is automatically arbitrated by the system and free resources are used to configure the hardware device. On legacy PCs, the information stored in the centralized Registry is used to notify the user of a potential resource conflict when configuring the peripheral. It is used also to perform device detection using the known resource information.

  With a Plug and Play PC, a user can easily install or connect Plug and Play devices to the system, letting the system automatically allocate hardware resources with no user intervention. For example, by simply plugging in a CD-ROM and sound card, a desktop PC can be easily turned into a multimedia playback system. The user simply plugs in the components, turns on the PC, and "plays" a video clip.

  Suppose the user wanted to install a new device on a legacy PC system. Further, suppose the new device requires an IRQ setting, and a legacy network card is already installed on the PC. Since the network card already uses IRQ 5, for example, the system tells the user that a device is already using IRQ 5 and that a different IRQ setting should be chosen. Device conflicts are a thing of the past.

- **PC systems can be designed with new features.**

  With warm-docking capabilities, for example, a businessperson could remove a portable PC from the docking station while the PC was still running, and go to a meeting. The portable PC would automatically reconfigure to accommodate the absence of the network card and large disk drive. Another example of this is an infrared (IR)-enabled subnotebook that would automatically recognize, install, and configure an IR-enabled printer when the user walked into the printer room.

Plug and Play Support in Chicago

As set forth by the industry initiative, the Plug and Play specifications are designed to be implementation-independent, and are not tied to a specific operating system. It is up to the operating system vendor to define the level of support the system will provide for making the PC easier to use.

Chicago was designed and built from the ground up with Plug and Play support in mind, and therefore provides a very rich implementation of Plug and Play functionality throughout every component of its design. With Chicago, configuration of hardware resources is greatly simplified over legacy configuration techniques—it just works.

Plug and Play in Chicago makes PCs even easier to use and supports both existing market requirements and future PC growth to deliver the following:

- **Compatibility with legacy hardware**

  With over 140 million MS-DOS or Windows–based PCs used throughout the world, providing compatibility with existing (or "legacy") hardware was a given requirement. The benefit of compatibility with existing hardware ensures support for Chicago and support for new Plug and Play peripherals does not require the purchase of completely new hardware.

- **Automatic installation and configuration of Plug and Play devices**

  This means that initial PC configuration is automatic. With Plug and Play, users no longer need to configure their system and make system-resource assignments. These assignments (including those for IRQs, I/O and DMA addresses, and memory) are handled by the BIOS and operating system, thus avoiding configuration conflicts. Installation and configuration of add-on devices and peripherals is also automatic.

- **Dynamic operating environment to support mobile computing environments**

  This functionality brings out the real power of the Plug and Play architecture, and sets Chicago apart from other operating system implementations of Plug and Play functionality. Dynamic Plug and Play properties in Chicago include support for:

  - Hot docking and undocking of mobile computers to change the state of the system dynamically
  - Hot plugging and unplugging of Plug and Play devices on the fly
  - "Dynaload drivers" where the operating system loads drivers for devices that are present and removes drivers from memory when the device is no longer available
  - Unified messaging for mechanism for notifying other operating system components and applications about changes to the state of the system dynamically

  Users of Chicago will be able to reconfigure their computer on the fly and have the changes take affect immediately, without rebooting the PC.

- **Simplified device driver development by using a universal driver model**

To simplify device driver development support for an IHV’s hardware device, Chicago incorporates the use of a universal driver model throughout various components in the system. Windows 3.1 supported a universal driver model for printer drivers, but Chicago provides this support for more areas including communications drivers, display adapter drivers, mouse drivers, and disk device drivers. The universal driver model ensures that it’s easy for IHVs to write peripheral drivers, thus providing for more Plug and Play devices available on the market.

- **An open and extensible architecture to support new technologies**

  The Plug and Play implementation in Chicago must be flexible and extensible enough to support future technologies as they emerge on the market The Plug and Play Initiative will spur the creation of new and innovative technologies, and Chicago will deliver this support.

- **Availability of configuration information for simplified systems management**

  This level of information sharing helps not only the solving of configuration problems for an end-user, but also the supportability and manageability of PCs within a corporate environment which may have hundreds or thousands of PCs. Through the use of the Registry, configuration information is easily available to the system and applications, and access to the information is made available to both locally and remotely.

Additional information about the Plug and Play capabilities in Chicago is discussed in the following sections.

Benefits of Plug and Play with Chicago

Plug and Play will be of enormous benefit to the user. No longer will the user be required to manually set jumpers and switches to redirect IRQs, DMA channels, or I/O port addresses. This will save the user’s time and will also save OEMs and IHVs the expense of supporting large numbers of user service calls related to these configurations.

Plug and Play is designed so that adding a device, either permanently or dynamically, requires nothing more than taking it out of the box and plugging it in. The PC seamlessly adjusts to the new configuration.

- **Users need not concern themselves with the inner workings of Plug and Play — it just works**.

  The Plug and Play specifications define how the various hardware devices, software drivers, and operating system components interact. At the level where the user interacts with the PC, the PC simply works. Plug and Play reduces the time users spend on technical problems and increases their productivity and satisfaction with PCs.

- **Plug and Play also benefits users who install Plug and Play devices into older, legacy PC systems**.

  Components using the Plug and Play architecture are able to accommodate the lack of device-reporting mechanisms in non - Plug and Play devices. Information about these devices is stored centrally in the Registry, and devices that cannot be reconfigured by the software receive first priority when resources are allocated.

- **Plug and Play makes it easier to manage and support PC configurations.**

  This is because many procedures that were once done manually—such as setting IRQ lines, figuring out what the right jumper settings are, and installing the correct device drivers—are now performed by the Plug and Play PC system. Problems that users used to encounter with non - Plug and Play PC systems generated a tremendous support burden. Customer frustration with the configuration process reduced demand for add-on and upgrade products. For businesses, the high cost of supporting PCs inhibited increased use of PCs in the workplace and diverted information systems personnel from focusing on using computer technology to solve business problems.

Improved Device Support

Chicago features improved support for hardware devices and peripherals including disk devices, video display adapters, mice, modems, and printers. In Windows 3.1 device drivers were, for the most part, monolithic and are complex for device manufacturers to develop. Windows 3.1 simplified printer driver development by using a mini-driver architecture, which provides printer device-independent code in a universal driver written by Microsoft, and device-dependent code that communicates directly with the printer written by the Independent Hardware Vendor (IHV). The mini-driver architecture increased the stability of the driver support for the printer and decreased the amount of time needed for a printer manufacturer to develop driver support for a new printer. While it is still possible to write monolithic drivers in Chicago, we recommend that IHVs use the mini-driver model because of the advantages it provides.

Chicago Device Driver Philosophy

Chicago extends the mini-driver architecture for printer drivers used in Windows 3.1 to the architecture for drivers of other system components. The driver philosophy that Chicago uses is based upon a mini-driver/mini-port layered model that provides the following benefits:

- **Leverages IHVs hardware knowledge**

  IHVs know their hardware. They understand the various I/O mechanisms that the hardware supports, and they know the commands that the hardware device will respond to. The mini-driver model allows the IHV to implement the device-dependent portion of the code used to interact with the hardware device.

- **Leverages Microsoft Windows knowledge**

  Microsoft developed the universal driver code, which is the layer of code that sits between the API layer of device interaction (as used by other Windows-components) and the device-dependent code that controls the device. The development team that wrote the Windows components above the API layer understands the mechanisms available from the operating system for interacting with the code. This leverages Microsoft’s knowledge of the operating system, with the IHVs knowledge of their hardware.

- **Increases system stability and reliability**

  Since the universal driver is the mechanism through which the Windows components communicate with the device, this components receives a high level of scrutiny and debugging. Through extensive use and testing, the universal driver code is made stable and reliable. Because the IHV no longer has to write the code that would be considered device-independent (as when they wrote monolithic drivers), the code required for driver-dependent functions for interacting with the hardware device is minimized. This reduces the complexity of the code that it necessary, and simplifies the driver development process. A simplified, less-complex driver will promise to be more stable and reliable than a traditional monolithic driver.

- **Increases forward compatibility**

  Forward compatibility is ensured by allowing the device-independent code to continue to evolve, and encapsulating the device-dependent code in a mini-driver. The mini-driver model would also simplify the extensibility of the driver an IHV would provide if new functionality was developed in the hardware device. The IHV would not need to completely rewrite the entire device driver, they would just add new functionality to the mini-driver (if even necessary).

- **Supports OEM/IHV innovation**

  The mini-driver model provides mechanisms for IHVs to add special device functionality support beyond what would be considered as a base set of required functionality. The mini-driver model doesn't require an IHV to sacrifice any flexibility to simplify the driver development process.

Chicago uses the mini-driver/mini-port layered model for components throughout the operating system, including printers, display devices, modems, communication devices, and mice.

Better Disk Device Support

In addition to providing compatibility with existing MS-DOS and Windows–based disk device drivers, Chicago provides better disk device support than is available under Windows 3.1. Chicago features a new block I/O subsystem that provides broader 32-bit disk device support as well as improved disk I/O performance. In addition, Chicago disk device drivers are compatibility with Windows NT mini-port drivers.

Chicago also enhances the disk device support provided in MS-DOS and Windows 3.1 to provide improved support in the following areas:

- **Support for large media using logical block addressing, including hard drives with greater than 1024 cylinders**

  Extensions to the Int 13h disk controller support are provided in the protected-mode disk handler drivers to support disks with cylinder numbers greater than 1024. (Windows 3.1 did not provide support for this in the 32-bit disk access drivers.)

- **Better support for removable media including electronic lock, unlock, and eject commands**

  Chicago better supports removable media devices and allows the system to lock or unlock the device to prevent the media from being removed prematurely. Chicago also supports an eject mechanism for devices that support it, so that users can use software control to eject media from a device (for example, new floppy drives that support software-based media ejection).

Support for IDE Drives and Controllers

Chicago provides improved support for IDE drive configurations. The enhanced support includes:

- **Support for large IDE disk drives**

  IDE drives are also emerging onto the market that support a logical block addressing (LBA) scheme that allows them to exceed the 1/2 gigabyte (528MB) size limitation. Support for large IDE disk drives as large as 137G will be provided by the Chicago operating system. While this support may be provided in real-mode today, Chicago provides this support in a protected-mode disk driver.

- **Support for Alternate IDE Controller**

  Chicago also allows the use of two IDE controllers in a PC, or the combination of an IDE controller in a laptop and an alternate controller in a laptop docking station (available, for example, in some Compaq laptop/docking station combination products). While this support may be provided in real-mode today, Chicago provides this supports in a protected-mode disk driver.

- **Support for IDE–based CD-ROM Drives**

  The majority of disk devices in personal computers today use an IDE-based hard disk controller. Adding a CD-ROM drive typically requires adding an additional controller card to provide either SCSI or a proprietary interface for connecting to the CD-ROM drive. A new crop of inexpensive CD-ROM drives that connect to IDE-compatible disk controllers are emerging onto the market, and Chicago recognizes and supports these devices.

Support for SCSI Devices and Controllers

Chicago provides great support for SCSI disk devices—something not available in Windows 3.1. The support in Chicago for SCSI devices includes:

- **Broad support for popular SCSI controllers**

Chicago includes 32-bit disk device drivers for popular SCSI controllers from manufacturers such as Adaptec™, Future Domain, Trantor, and UltraStor, providing great support right out of the box.

- **Compatibile with Windows NT mini-port drivers**

  Chicago supports the use of Windows NT mini-port SCSI drivers under Chicago without modification or recompiling. Compatibility with Windows NT-based mini-port drivers ensures broad device support for disk devices under Chicago, while simplifying the driver development efforts for hardware manufacturers.

- **ASPI/CAM compatibility for MS-DOS–based applications and drivers**

  Support for the Advanced SCSI Programming Interface (ASPI) and Common Access Method (CAM) allowing application and driver developers to submit I/O requests to SCSI devices is provided in Chicago. This will allow existing MS-DOS–based applications and drivers that use the ASPI or CAM specification to work properly under the Chicago operating system.

- **16-Bit and 32-Bit ASPI for Windows–based clients and applications**

  In addition to MS-DOS–based compatibility with ASPI, Chicago also includes 16-bit and 32-bit drivers to support Windows–based ASPI clients and applications.

Support for ESDI Controllers

Chicago provides 32-bit disk driver support for ESDI controllers in addition to supporting IDE and SCSI disk devices.

High-Speed Floppy Disk Driver

As with its hard disk controller support, Chicago also provide protected-mode support for communicating with floppy disk controllers. Chicago provides Int 13h hard disk controller support as 32-bit device drivers resulting in improved performance, stability, and robustness of the system. Chicago provides floppy disk controller support as a 32-bit device driver, and offers improved performance for file I/O to floppy disk drives, plus improved reliability of the system.

Users can now effectively format a diskette or copy files to/from a diskette while performing other tasks.

Better Display Adapter and Monitor Support

Video display adapter and monitor support in Chicago is another area that has received a lot of attention during the design phases of Chicago.

Summary of Improvements Over Windows 3.1

Chicago addresses many of the problems inherent in Windows 3.1 display drivers and provides enhanced functionality and easier setup and configuration. Benefits of the new display driver support in Chicago includes:

- More stable and reliable video display adapter drivers
- Many more video cards supported by drivers in the box
- A mini-driver architecture that makes it easier for IHVs to write video display drivers
- Support for new features including the ability to change video resolution on-the-fly without needing to restart Chicago
- Consistent and unified installation and configuration of display drivers and display properties such as colors, wallpaper patterns, and screen saver
- Image Color Matching support for device-independent color usage, which Microsoft worked in conjunction with Kodak to offer
- Support for new generation of hardware and device functionality such as Energy Star Monitors conforming to the VESA DPMS specification, and detection of monitor properties such as maximum resolution supported (with Plug and Play monitors)

Improved Driver Stability and Reliability

By using a mini-driver architecture for video display adapter drivers, Chicago better supports the range of products offered by IHVs and provides more stable and reliable drivers. Chicago provides a universal driver to support device-independent code and functionality normally handled by a monolithic video display driver, and supports device-dependent code in a display mini-driver. The mini-driver uses the Chicago graphics device independent bitmap (DIB) engine, providing a better mechanism for manipulating memory bitmaps (including improved performance).

Because the mini-drivers are simpler than a monolithic display driver, they are easier to write and to debug. Extensive testing on a less-complex driver results in better stability and reliability in the overall operating system.

Furthermore, to ensure broad display adapter device support in Chicago, Microsoft is developing many of the display drivers in-house with cooperation of all major display controller IHVs. The development teams at Microsoft are also working closely with IHVs to write additional display drivers, and assisting IHVs with optimizing their display drivers and doing performance tuning to enhance the speed at which information is displayed by the driver. The development effort will result in improved graphic performance over Windows 3.1 and native Windows 3.1 display drivers.

The use of the mini-driver architecture for display drivers in Chicago leverages the development experience that Microsoft has for writing fast, reliable graphics code, with the engineering experience of IHVs, allowing them to concentrate on delivering high-performance hardware accelerated display adapters.

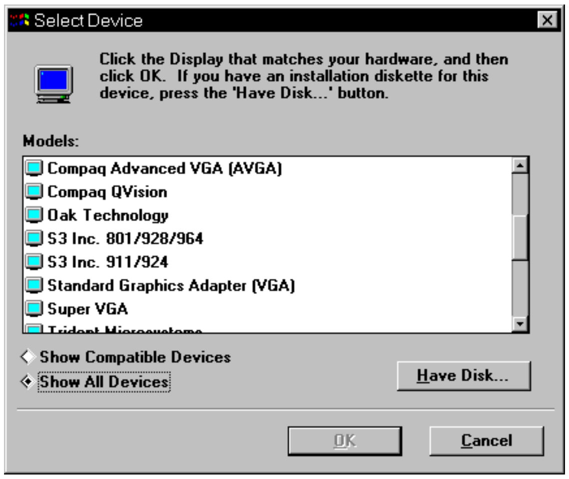
Improved Video Display Performance

In addition to more stable and reliable video display adapters in Chicago, display drivers also should benefit from improved performance. The mini-driver architecture for display drivers in Chicago is centered around a new 32-bit DIB engine that features 386/486 optimized code for fast, robust drawing for high resolution and frame buffer-based display adapters. The use of a universal driver to provide the device-independent display adapter support, instead of requiring each IHV to redesign this code, results in allowing base functionality to be optimized and thus benefits all mini-driver display drivers.

Support for More Video Display Adapters Than Windows 3.1

Setup in Chicago includes support for automatically detecting the video display adapter installed in the PC and installing the appropriate Chicago display driver. While Chicago supports the use of display device drivers written for use with Windows 3.1, Microsoft is working closely with IHVs to provide Chicago-specific display drivers that take advantage of new features and functionality available in Chicago. For example, efforts are on going to assist third-parties in implementing extensions to support plug and play detection, on-the-fly resolution changes, and re-architecting display drivers to leverage the mini-driver model. Microsoft is also developing display drivers for the top display adapters and chipsets available in the industry.

The display drivers listed in the Display section of the Select Device dialog box will provide a list of display adapters recognized in the Beta-1 release. This list will most likely change and grow between now and Chicago's final product release. We will continue to add support for more existing display adapters as well as new adapters that are still emerging onto the market. For display drivers not shipped in the box, the Windows Driver Library (WDL) will provide the distribution mechanism to support additional third-party drivers.


Select Device

Click the Display that matches your hardware, and then click OK. If you have an installation diskette for this device, press the 'Have Disk...' button.

Models:

- Compaq Advanced VGA (AVGA)
- Compaq QVision
- Oak Technology
- S3 Inc. 801/928/964
- S3 Inc. 911/924
- Standard Graphics Adapter (VGA)
- Super VGA

Figure 42. Select Device Dialog Box for Supported Display Adapters

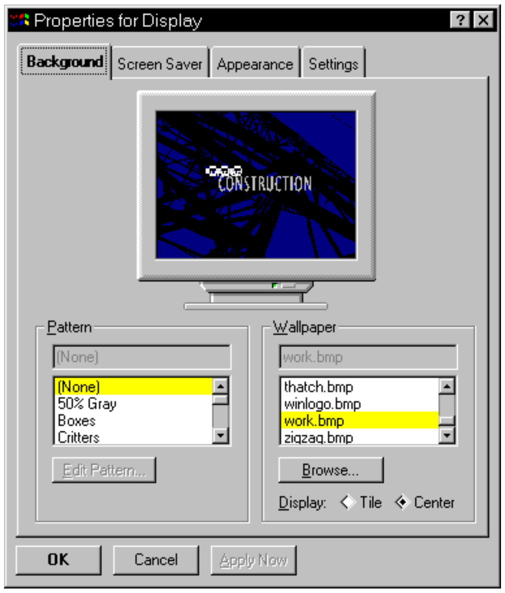
Robustness Improvements

The video drivers provided with Chicago are stringently tested to ensure greater reliability and stability than drivers for Windows 3.1.

In addition to a better quality of video drivers, Chicago includes mechanisms to ensure that bad or incompatible video drivers cannot prevent user from accessing the system. If a video driver fails to load or initialize when Chicago is started, Chicago defaults to the generic VGA video driver. This ensures that a user can get into Chicago to fix the system, given that driver configuration is handled through a graphical interface. Under Windows 3.1, a bad video driver would commonly result in returning the user back to an MS-DOS command prompt with no explanation about the failure.

New Control Panel Enhancements and Customization Properties

Chicago consolidates display properties into a common Display area in Control Panel, allowing easy customization the colors, wallpaper, screen saver, and display adapter settings from a single user interface. Access to display properties is as easy as selecting Control Panel from the Settings option from the Start button (or by using the secondary mouse button to click the desktop), to quickly present the display property sheet to the user.



Pattern: (None)

- (None)
- 50% Gray
- Boxes
- Critters

Wallpaper: work.bmp

- thatch.bmp
- winlogo.bmp
- work.bmp
- zigzag.bmp

Display: Tile [✓ Center]

Figure 43. Properties for Display Dialog Box

Through the new consolidated display properties, users now have the ability to:

- Change video resolution on-the-fly when using display drivers and display adapters that support this functionality.
- Change color schemes or window properties for displaying text in title bars or menus such as font face to use, styles of fonts to use including bold or italic, and sizes of fonts to use, providing more flexibility and levels of customization than Windows 3.1.
- See the appearance of display changes modeled on-screen before the changes are applied. This capability has been referred to as What You See *Before* You Get It (WYSBYGI).

The work done for consolidated display properties is a further example of how Chicago is making it easier for users to use and customize their environment.

Image Color Matching Support

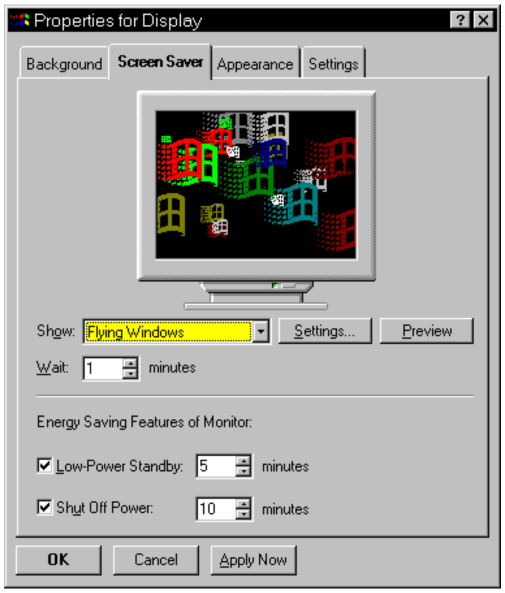
Chicago provides image color matching (ICM) support for mapping colors displayed on-screen and colors generated on output devices to provide consistent output. See the discussion of ICM support in the Printing section of this guide.

Energy Star Monitor Support

Energy Star is an Environmental Protection Agency (EPA)-inspired effort to develop computer hardware and peripherals that conserve power while in idle states. This ides is similar to the standby-mode commonly implemented in laptop computers to save power.

In a PC system, the video display monitor is typically one of the power-hungry components. Manufacturers of newer display monitors have incorporated energy-saving features into their monitors based on the VESA Display Power Management Signaling (DPMS) specification. Through signals from a video display adapter, it is possible under software control to be able to place the monitor in a standby mode, or even turn it off completely, thus reducing the power it uses when inactive.

User today typically use a screen savers to prevent burn-in of a monitor image. Chicago extends this mechanism to provide a time delay setting allowing the user to put the display monitor in a low-power standby mode, as well as a delay setting to turn the monitor off completely. Figure 44 below shows the delay settings that a user may specify to enable this capability.



Show: Flying Windows

Wait: 1 minutes

Energy Saving Features of Monitor:

☑ Low-Power Standby: 5 minutes

☑ Shut Off Power: 10 minutes

Figure 44. Screen Saver Settings for Energy Saving Monitor Features

For example, a user may want to set options to display a specific screen saver after 5 minutes of inactivity, to set the PC to standby after the screen saver has displayed for 10 minutes, and turn off the monitor after 15 minutes of standby.

To take advantage of the Energy Star power-consumption mechanisms, it is necessary to have both a display adapter and a monitor that meets the Energy Star specifications. It is also necessary for the video display driver to support the extensions necessary to control the monitor device. Several manufacturers are presently shipping monitors that are designed to support the Energy Star goals.

Better Mouse and Pointing Device Support

As with other device drivers, Chicago's mini-driver architecture simplifies mouse driver development and improves virtualization in a protected-mode mouse driver to better support MS-DOS–based applications in the Windows environment.

Summary of Improvements over Windows 3.1

Mouse support in Chicago results in the following improvements over Windows 3.1:

- Provides smooth, reliable input support through the use of protected-mode drivers
- Supports more devices by making it easier for IHVs to write drivers, and supports a mini-driver architecture model
- Makes mouse and pointing devices easy to install and use by supporting Plug and Play
- Implements mouse driver functionality in a single driver, and eliminates the need to use MS-DOS–based mouse drivers (increasing robustness and saving conventional memory)

Improved Windows Mouse Driver

Windows 3.1 provided support for using the system mouse in an MS-DOS–based application if the application was run within a window. However, support for using a mouse in full-screen mode required an MS-DOS–based mouse driver TSR to be loaded prior to starting Windows.

Chicago provides mouse support as a protected-mode VxD and eliminates the need to load an MS-DOS–based mouse driver. Better virtualization of mouse interrupt services, which allow protected-mode Windows–based mouse driver to provide mouse support for Windows–based applications, MS-DOS–based applications running in a window, *and* MS-DOS–based applications running in full-screen mode. The improvements in this area result in a zero conventional memory footprint for mouse support in the Chicago environment.

In addition to better mouse services, Chicago improves the device support to allow the use of serial ports COM1 through COM4 on which to connect a mouse or other pointing device.

Mouse Control Panel Enhancements

Chicago consolidates mouse configuration and customization support into a single Control Panel icon. Windows 3.1 provided rudimentary support for configuring a mouse as part of the Mouse option in Control Panel, and provided more flexible mouse settings in a separate driver-specific applet.

Control over mouse customization options is supported in Control Panel, and uses a tabbed dialog for providing easy access to the different possible settings. Mouse settings accessible through the new interface includes setting the behavior of the mouse buttons, and the behavior of the mouse pointer.

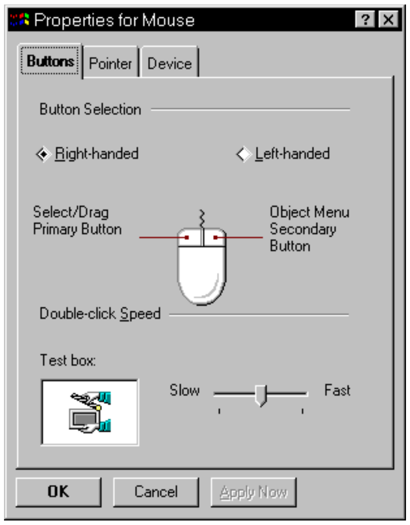


Figure 45. Properties for Mouse Dialog Box

Try It!

Mouse

To see how the improved device support present in Chicago will result in broader support for a broader base of hardware and peripherals, you've got to try it!

Floppy Disk and Multitasking Performance

To see the improvements made in the floppy disk driver, try to perform some common tasks under Chicago while you are formatting a floppy disk or copying files to a diskette. For example, try navigating through the shell or launching another application. Perform the same tasks under Windows 3.1 to compare the different multitasking behavior.

Single Mouse Driver

To see the improvements made in mouse driver support to reduce the conventional memory used as facilitated by a single system mouse driver, remove the real-mode mouse driver from your CONFIG.SYS or AUTOEXEC.BAT and (after restarting your PC), run an MS-DOS–based application that supports the use of a mouse. Use an application such as Edit and try the MS-DOS–based application both in a window and full-screen. Note that the mouse is available in both modes, and use the **mem /c** command at the MS-DOS prompt to verify that the mouse driver is not loaded into real mode.

Networking

Windows desktops are being connected to corporate networks at a steadily increasing rate, along with it are growing demands for better network integration, improved network and system management capabilities, and better network performance and reliability as more business critical functions rely on the PC network. As a consequence of these demands, companies are faced with increased costs to run PC networks and are investing in tools and staff to meet the challenge of day to day management of their growing corporate PC networks. Chicago is the version of Windows constructed to address the needs of the corporate network administrators with a well-integrated, high-performance, manageable 32-bit network architecture.

Chicago is also designed to address the needs of the Windows user, making access to and control of the network consistent, and easier to use through many enhancements in the Windows user interface making network browsing and printing much easier to use. In addition, Chicago is designed to address users mobility needs both roving on the corporate network, as well as enabling remote access to the network from portable PCs.

Given the size of customer's current investments in both Windows and their PC network infrastructure, one overriding goal for Chicago networking is compatibility. Compatibility starts by ensuring continued support for existing real-mode networking support, as well as making Chicago's new 32-bit protected-mode components very compatible with the 16-bit MS-DOS applications and device drivers and the 16-bit Windows applications and DLLs that customers use today.

This section of the *Reviewer's Guide* will introduce you to the 32-bit, protected-mode networking architecture built into Chicago and will show you how it provides well integrated network support, manageability, improved performance, user-level network security and remote access to the network. The Chicago networking discussion is structured as follows:

- **Easier Networking with Chicago**. Summarizes the key features and concepts in Chicago that make networking much easier to implement and use.
- **Chicago Network Architecture**. Details of the internals of Chicago's new 32-bit protected-mode networking infrastructure.
- **Managing Chicago Systems**. Outlines the support built into Chicago to enable both System Management and Network Management.

Summary of Improvements over Windows 3.1 and Windows for Workgroups 3.11:

The primary improvements in networking for Chicago are:

- A robust, open, high-performance 32-bit network architecture—32-bit network client software, 32-bit file and printer sharing software, 32-bit network protocols, and 32-bit network card drivers
- Support for using multiple redirectors, multiple protocols and network card device drivers simultaneously to facilitate integrating the desktop into a heterogeneous network environment
- Support for industry standard connectivity and systems management solutions including TCP/IP, IPX, SNMP, and DMI
- Great integration with Novell NetWare including high-performance, 32-bit protect-mode NetWare-compatible client software for connecting to NetWare 3.x and 4.x servers, and peer sharing for Netware environments
- Great integration with Windows NT Advanced Server to support a powerful client/server solution
- Built-in support for systems management, including the ability to remotely administer, monitor, and view the configuration of PCs over the network
- Improved remote network access support providing remote access to Microsoft Networking servers, Novell NetWare servers, and UNIX servers. Support for remote protocols such as PPP and SLIP is provided.
- Improved network printing, making it easier for users to connect and configure printers in network environments.

Easier networking with Chicago

This section summarizes the key features and concepts in Chicago that make networking much easier to implement and use.

Chicago Provides great Novell NetWare Integration

Chicago has built-in support for two networks—Microsoft and Novell NetWare networks. (Built-in support for Novell NetWare is new for Chicago.) Installation of support for one or both networks is as simple as a clicking the Chicago Setup program or the Network Setup icon in Control Panel. Both the Windows Network and the Microsoft Client for NetWare are implemented as high-performance, high-reliability 32-bit protected-mode components.

Microsoft Client for NetWare

The Microsoft Client for NetWare for Chicago provides interoperability for NetWare 3.*x* and 4.*x* servers. Chicago systems can use all NetWare server services, browsing NetWare servers, connecting to servers, queue print jobs either using the Chicago network user interface, or using Novell's NetWare command line utilities. In fact Chicago's Microsoft Client for NetWare will even run "TSR clean" NetWare login scripts. In addition, Chicago provides continued support for Novell NetWare real-mode components. This means the NetWare 3.*x* NetX shell and the NetWare 4.*x* VLM shell are both supported by Chicago.

Microsoft File and Print Sharing for NetWare

Chicago also provides NetWare compatible peer services for file and print sharing that feature user-level security by implementing a “pass through” security link to an existing Novell NetWare server. Chicago’s doesn’t introduce a new security scheme; rather, it fully leverages the existing user-level security built into NetWare’s bindery or NDS.

The Microsoft Network support provides full interoperability with other Chicago PCs, and PCs running Windows for Workgroups, Windows NT, Windows NT Advanced Server, LAN Manager, and any other Microsoft-compatible servers. Chicago includes support for both client access, and peer services capabilities on a Microsoft Network.

Additionally, other network servers and services will be provided by third parties, for example Artisoft®, Banyan®, DEC®, and SunSelect will provide Chicago support for their respective network servers.

Chicago is the “Well-Connected Client” Operating System

Today’s networks are heterogeneous, and becoming even more connected. Companies are linking their Windows PCs to multiple PC network servers, mainframe and mini-computer host systems, UNIX machines, and even a variety of services like the Internet. The desktop operating system must meet this challenge and provide support for often very disparate connectivity needs on the network. Today’s desktop operating systems do not provide the necessary support for running multiple network clients simultaneously. Chicago has been explicitly designed with multiple network support as a key design goal.

Integrated networking support is a key focus of Chicago’s design, it’s now much easier to install and manage support for a single network or even multiple networks simultaneously using Chicago. Building upon the support in Windows for Workgroups 3.11, which was capable of supporting up to two networks, Chicago has the capability to simultaneously support two or more networks using the Chicago Network Provider Interface. This interface defines a set of APIs used by Chicago to access the network for things like logging on to the server, browsing servers, connecting to servers, printing, and so on. Installing network provider support is simple—it’s done via the Network Setup icon in the Control Panel, or when first installing Chicago from the Network Setup dialog box. This means that a Chicago desktop can run client support for NetWare, Windows NT Advanced Server, Banyan and Sun NFS simultaneously.

Chicago Makes Using the Network Easy as “Point and Click”

For users, running one network client can be confusing and multiple network support is nearly unmanageable. Each server has its own set of unique client-side utilities and commands that are often difficult to remember and use. When the desktop PC has multiple network support loaded, the user is now faced with minimally twice the number of commands and utilities to remember and may now have to remember multiple passwords to access network resources. Chicago’s easy to use Network Neighborhood user interface make it easier for users to perform common network operations on very disparate servers from a Chicago. First, it’s now possible for network manager to establish one password to log the user into the Chicago PC and any network resources or services that they are entitled access to. These services could include email, group scheduling applications, dial in support or database access.

Additionally, common network actions like browsing servers, managing connections and printing are all done identically through the Chicago user interface regardless of the type of server Chicago is connected to. This means a user can easily locate, connect, and start a print job on a NetWare print server as easily as they can for a printer attached to a Windows NT Advanced Server. All the common network actions can be accomplished visually, using the mouse to navigate through the network resources, manage connections, and so on. The user isn’t required to memorize any new network commands. For both Windows Networks and Novell NetWare, the user can run the corresponding command line utilities as well. This ongoing backward compatibility may be necessary to support batch files currently in use, or to help manage the transition period moving to the Chicago environment.

Lastly, the Network Neighborhood helps to manage the complexity of the network by showing it from the user’s perspective. That is, it will show only what the user is interested in seeing. When the user initially opens the Network Neighborhood the window will only contain the servers that the user has logged into, or servers that the user most frequently connects to. This context-sensitive view the network thus reduces the number of network resources that the user initially encounters to a more manageable number of objects. For Windows NT domains and NetWare 3.*x*, the network context presented is the “login server” and any other connected servers, for NetWare 4.*x* the Organization Unit (OU) context for the user is the context presented as well as any other connected servers.

For more in depth discussion of the Network Neighborhood and the Chicago user interface, see to the “The Chicago User Interface” section in this *Reviewer’s Guide*.

Chicago Makes Mobile Network Support Easier

Two features in Chicago make connecting to a network easier for mobile PC users—Plug and Play and Remote Access.

- **Plug and Play.** Chicago's Plug and Play solves several problems that face mobile PC users. are faced with a variety of challenges to keep their PCs running smoothly today. Mobile users no longer have to maintain multiple configurations (such as desktop and portable configurations)—Chicago recognizes when they add or remove peripherals, such as when they remove a network card and add a modem for dial-in network access. By supporting hot and warm docking, users no longer have to reboot their systems each time they make a change to the configuration. In addition, Chicago has built-in Card and Socket Services which allow for hot removal and insertion of PCMCIA cards, including network cards.

  Finally, Chicago's network Plug and Play support includes application-level support. An application that is network-aware understands whether the network is available or not. If network adapter is removed, the application automatically put itself into "offline" mode to allow the user to continue to work, or it shuts down gracefully.

- **Remote Access.** Maintaining data access to their corporate network while working in a remote location is another challenge for mobile users. Currently, several solutions for dialing-in to the corporate network exist. However, most of these solutions are not well integrated with Windows, requiring a different set of tools. Chicago's Remote Network Access client provides modular support for multiple dial-in providers, including Windows NT RAS servers and NetWare. It also supports several protocols, including NetBEUI, IPX/SPX and TCP/IP via PPP. Support for dial-in can also be offered by third parties, including Shiva, using the modular architecture of Chicago's Remote Access client.

Chicago Client: Designed for Manageability

Many corporations have rapidly growing networks, networks that in some cases run worldwide. Keeping the networks and ever increasing number of systems connected to the networks running at peak performance is a challenge for both end users and network managers. These corporations are beginning to deploy network and desktop management tools to help them meet this challenge. Chicago has built-in network and system management instrumentation to enable current and future management tools to remotely monitor, query and configure Chicago PCs. Using these tools, network managers will be able to quickly inventory software and hardware used on their network. Working from a Chicago PC, network managers can remotely diagnose and reconfigure Chicago systems as well as remotely monitor system and network performance on a Chicago PC. The following key components make Chicago very manageable:

- **SNMP Agent.** Chicago incorporates an agent that implements the Simple Network Management Protocol (SNMP). This agent complies to the Internet SNMP specification, responding to queries and sending notifications of events that take place on the PC to an SNMP console. The SNMP console allows a network manager to remotely monitor and manage the Chicago PC. Events can be managed from a central SNMP management console.

- **SNMP MIB, MIB 2.** The SNMP MIB describes what information about the system is available to the SNMP console. Chicago includes the MIB-II which describes the Microsoft TCP/IP protocol, and allows information about the protocol stack to be communicated back to the management console. For example, the management console can query the MIB-II for the IP address, the name of the user at this IP address or IP routing information.
- **DMI Agent.** Chicago offers a DMI agent soon after final Chicago release. DMI applications offer cross-platform desktop management capabilities. Support of the DMI agent is built on top of Chicago's Registry. The DMI specification is still being developed, Microsoft as a founding member of the DMTF will follow its development.

- **Registry-based System Management.** Central to Chicago's operation is the Registry. Similar in design to the Registry found in Windows NT, it replaces the many .INI files previously used by Windows and Windows applications. The Registry contains information used by Chicago that describes the hardware configuration of the PC, preferences defined by the user and application specific information. The Registry is a database containing keys, and values. For example, HKEY_USER_NAME defines the key for the user's name. The name "Fred Smith" is the values associated with this key. There also exist a special category of keys called Dynamic Keys. These keys are memory resident, and can contain frequently changing data updated by system components, device drivers or applications. For example, the number of packets sent per second could be registered by the network adapter device driver.

  The Registry consists of three components—SYSTEM.DAT which describes the PC configuration and computer-specific application information, USER.DAT that defines user preferences and user specific application information, and POLICIES.DAT which defines the "corporate policies" relating to either previous component. Each component is a file that resides on the PC or on a network server. The Registry is remotely accessible via an RPC based interface. The APIs used to access the Registry both locally and remotely are the Win32 Registry APIs.

Chicago Management Tools

There are several tools for Chicago that make managing the system or the network much easier for a PC or network manager. These tools include:

- **Registry Editor.** It allows local or remote editing of the Chicago Registry
- **System Policy Editor.** This is used by network managers to set "policy" over-rides on Registry entries per user or per group and creates the POLICIES.DAT component of the Registry. This tool contains a superset of Windows for Workgroups' "admincfg" tools settings.
- **Performance Monitor.** It allows you to locally or remotely view the performance of the various i/o components of the local system or remote PC. monitor the file system, the network components, or data from the network card. the data is updated dynamically using the Registry "dynamic keys."
- **NetWatcher.** This allows you to locally or remotely view the network connections of Chicago peer services.

Easier to Setup and Install

PC and network managers faced several challenges when installing Windows in the past. Some network managers installed Windows on the network for later installation onto users' PCs, or to run Windows from a network server. In the first case, the network manager had to decide on an approach for a number of variables—making the process appear transparent to the user, rolling out Windows using a "push" installation, using specific settings for different categories of users, and updating these configurations when either Windows, Windows applications or device driver updates are available.

Running Windows from a network server, network managers had to manage variables such as having local swapping files and some local .INIs and applications, allow user-level configurations, how to support disparate hardware configurations, and handling the roving user on the network.

Chicago addresses several elements of these problems with an improved Setup utility and the previously discussed Chicago Registry. The new Setup streamlines the installation of Windows on a network server for either installation from the server, or running Chicago from a server.

Running Chicago from a server becomes much simpler largely due to the new Chicago Registry. The Registry is a centralized database of all hardware, software and user information, hence, is easy to maintain remote on the server. Contrast this to the state of configuration today with CONFIG.SYS, AUTOEXEC.BAT and the myriad of Windows and Windows applications .INI files. In addition, the separation of hardware configuration from the user profiles means that users can rove on the network, their preferences will follow them from PC to PC regardless of the hardware configuration they're currently running on.

Chicago Network Architecture

The Network Architecture in Chicago radically updates the level of network support and integration that existed in Windows 3.1. The key design points of Chicago's Networking Architecture are:

- **Fast, 32-bit VXDs**. Chicago's networking components are built as Windows Virtual Device Drivers (VXDs). VXDs are 32-bit, and have no conventional memory footprint. In addition, since the operating system and the device drivers are all running in protected-mode, network I/O performance is 50 to 200% faster than Windows 3.1 because there's no more overhead for mode switching between protected and real-mode operation.
- **Reliable**. Since Chicago's networking components run in protected-mode and are designed to a well-defined set of interfaces, they are more reliable than real-mode network components. Today's real-mode network components may conflict in memory or attempt to exclusively chain the same set of interrupts, this commonly leads to system hangs or error conditions. Chicago arbitrates the hardware resource allocation, hence these errors won't occur with protected-mode network components.
- **Modular, Open Design**. Chicago's Network Architecture is highly modular, which includes a new Network Provider interface, an Installable File System (IFS) interface, and an enhanced version of Network Driver Interface Specification (NDIS) version 3.1 which has been enhanced for Plug and Play support. The specifications are available for all three aforementioned interfaces for third party network vendors.
- **Multiple Network Support**. Chicago is designed to accept multiple Network Providers, multiple network redirectors written to the IFS interface and multiple NDIS drivers as needed. This means it is possible to run Microsoft Network and Novell NetWare client support simultaneously.
- **Multiple Protocol Support**. Chicago's Protocol Manager supports loading multiple transport protocols. Protocol Manager is one of the NDIS components, and makes it enables Microsoft and third parties to independently author protocol stacks for Chicago that coexist well. Chicago includes built-in support for IPX/SPX, TCP/IP and NetBEUI.
- **Plug and Play Enabled** The whole of Chicago's Networking components are designed for dynamic Plug and Play operation. For example, when a PCMCIA network adapter is inserted, the NDIS 3.1 network card driver is automatically loaded, and the network is available. Alternately, if the PCMCIA network card is removed, or the network cable is removed, Chicago will not hang as many real-mode networks do, but will notify any applications using the network that it's no longer available and will continue to run.

Figure 46 shows an overview the network architecture built into Chicago. The following sections describe key aspects of this architecture, including the Network Provider Interface, the Installable File System, and NDIS 3.1.

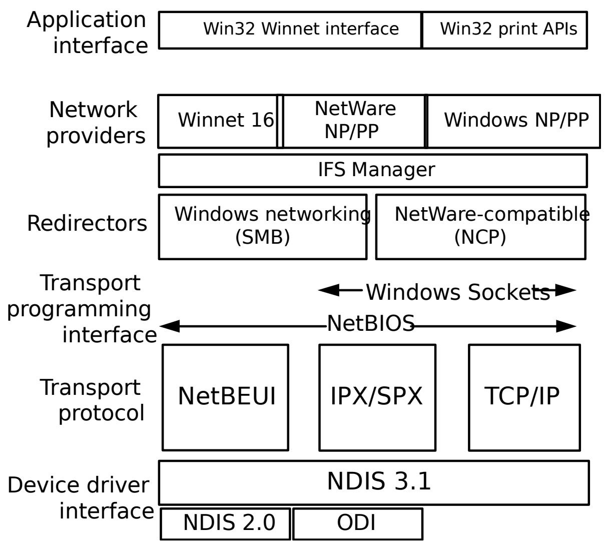


Figure 46. Diagram of Chicago's Layered Network Architecture

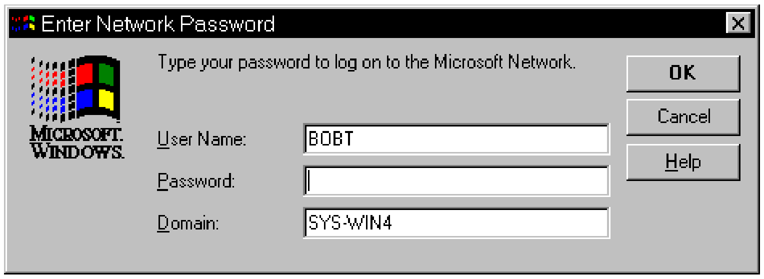
Network Provider Interface: Concurrent Support for Multiple Network Servers

Chicago has an open, modular Network Provider Interface (NPI) to allow multiple network support to be installed in Chicago simultaneously. NPI enables Microsoft, or any third party network provider to integrate varied network services seamlessly into Chicago. Key benefits of the NPI are:

- An open interface that allows any network vendor to supply tightly integrated support for their network servers for Chicago
- All supported networks are identically accessed and managed through the Chicago Network Neighborhood user interface

The NPI abstracts the network services for the Chicago user interface components, as well as the various Chicago network and desktop management components. The Network Provider Interface consists of two parts—the network provider API and the network providers. The network provider API is a single, well defined set of API used by Chicago to request network services such as browse servers, connect and disconnect to servers, queue a print job, and so on. These requests are then passed to the network providers. The network provider layer sits below the API layer, and provides the needed network services to honor Chicago's request for network specific services. Conceptually, this model is similar to the design of Chicago's various device driver interfaces, a well defined set of interfaces used by the operating system, and the services provided by a device driver often written by a third party that honors the request.

The most apparent abstraction of the various network services provided by the Network Provider Interface is the Chicago system login. Each Network Provider can provide a unique login dialog box to suit the needs of the network servers security model. For example, the login dialog box below is for logging in to a Windows NT Advanced Server domain:



Enter Network Password

Type your password to log on to the Microsoft Network.

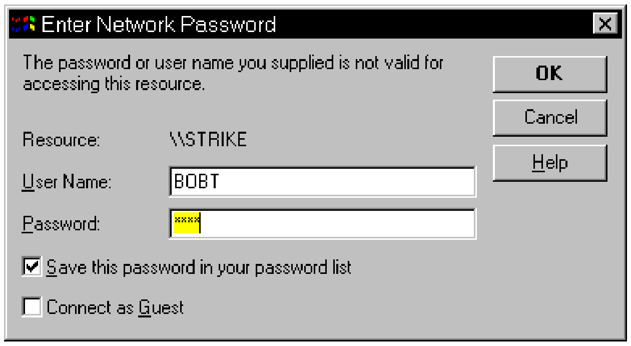
User Name: BOBT

Password:

Domain: SYS-WIN4

Figure 47. Network Logon Dialog Box for Windows NT Advanced Server Domain

Note that the dialog box below for logging in to a Novell NetWare 3.*x* server offers additional information to allow users to logon as GUEST. This dialog box is invoked when a user first accesses a NetWare server.



Enter Network Password

The password or user name you supplied is not valid for accessing this resource.

Resource: \\STRIKE

User Name: BOBT

Password: ****

[✓] Save this password in your password list

[ ] Connect as Guest

Figure 48. Network Logon Dialog Box for Novell NetWare 3.x

To complete this example, once the login is validated against the requested server, this is passed back to Chicago. Chicago can then use this password as the "Master Key" and unlock and system or network resources that are linked to the Master Key password validation. In this fashion, it's possible for Chicago to accommodate various ways that network servers provide their services, yet still offer the user a very consistent user interface.

Another examples of user visible support from the Network Provider occurs when specifying server name strings. For example, Microsoft compatible networks use the Universal Naming Convention (UNC) which appears in this form:

\\*server-name**share-name*

However, NetWare servers are specified in this form:

server-name/*volume-name*:*directory-name*

The respective Network Providers will correctly parse the syntax of their server name strings. This means that users who are accustomed to using the NetWare server syntax can type the NetWare server syntax string wherever required by the Chicago user interface to access NetWare server resources.

Installable File System: Support for Multiple Network Redirectors

The Installable File System (IFS) interface built into Chicago is a well defined set of APIs that are used to implement all file systems in the operating system including the following; VFAT (32-bit FAT) and CD-ROM file systems. The Chicago IFS implementation is functionally similar to the IFS implementations on Windows for Workgroups and Windows NT. For networking, the IFS is used to implement network redirectors. The IFS interfaces are documented, and are meant to be used by vendors of network servers to implement their Chicago redirector. The IFS offers a number of key benefits for Chicago network redirectors:

- Designed for multiple redirector support
- Increased reliability, the IFS model arbitrates resource requests, removing the source of many real-mode redirector conflicts
- Improved performance, network redirectors will benefit from the unified IFS cache making available client side network redirector caching

The IFS consists of a set of file system APIs and loadable File System Drivers (FSDs), multiple FSDs can be resident in the system simultaneously. The FSDs provide the logic necessary for the file system to provide a consistent logical view of devices, and arbitrates access, update and control of devices consisting of very different physical media types. For network redirectors, the FSD provides mechanisms to locate, open, read, write and delete files, as well as services like named pipes and mailslots.

To illustrate the flow of control, take as an example opening a file that is actually a link to a file on a server on the desktop. The user double-clicks the icon, then Chicago's shell parses the link and determines that the file is a network object. The shell passes the filename to the NPI, which may re-establish the network connection to the server on which the object resides, if required. The NPI then in turn calls the network redirector to open the file on the file server. The network redirector translates the file request into a request formatted for the specified network file server, transmits the request to the server via its link through the NDIS layer, and returns a handle to the open file back up to the NPI and the shell.

Both Microsoft supplied Microsoft Network Client and the Novell Compatible Client are implemented as IFS FSDs.

NDIS 3.1: Multiple Protocol Support and ODI Support

The Network Driver Interface Specification (NDIS) version 3.1 is a superset of the NDIS 3.0 functionality that exists for Windows NT and Windows for Workgroups 3.11. NDIS 3.1 has enhancements for Chicago in two key areas:

- Plug and Play enhancements to the Protocol Manager and Media Access Control (MAC) layer that enables network drivers to be dynamically loaded and unloaded
- A new NDIS mini-driver model, the Chicago mini-drivers are binary compatible with Windows NT Daytona's mini-driver implementation.

Upgrading an NDIS 3.0 driver to NDIS 3.1 is very straight-forward. (For example, in some cases the changes have taken one hour for Microsoft engineers to update an NDIS driver source code.) Instead of making this type of upgrade, vendors may instead choose to provide a mini-driver. As noted previously, the primary changes to the NDIS model were extensions for Plug and Play support.

The mini-driver model dramatically decreases the amount of code that a network adapter vendor must write. Conceptually, this model is similar to the driver models implemented for printers, disk drivers, and display drivers. Essentially the mini-driver divides the existing NDIS Media Access Control (MAC) layer into two halves. The mini-driver half implements only the code that is specific to the network adapter card. These include specific implementation details like establishing communications with the card, turning on and off electrical isolation (if implemented) for Plug and Play, doing media detection and enabling any value added features the card may contain. The mini-driver is wed to the NDIS wrapper, which implements the other half of the MAC functionality. This contains the code that remains "common" to all NDIS drivers. In prior releases of NDIS, each MAC carried all this redundant code, hence, the mini-drivers are much smaller than existing NDIS 3.0 MACs, roughly 40% smaller in size. NDIS mini-drivers developed for either Windows Chicago or Windows NT Daytona are binary compatible.

An NDIS 3.1 stack is composed of three component parts—the protocol, the MAC or mini-port, and the mini-port wrapper. NDIS contains the protocol manager which loads and unloads the protocol. This manager can manage multiple protocols loaded simultaneously. Just below is either the MAC or mini-driver wrapper, if using mini-drivers. Multiple MACs or mini-drivers can be loaded in systems that have multiple network adapter cards loaded. Finally, the mini-port wrapper layer below the mini-port does a mapping of Windows NT Hardware Abstraction Layer (HAL) layer APIs for I/O. This mini-port wrapper layer is very thin, since Chicago can always assume that it's being run on an Intel architecture.

Novell NetWare Integration

Chicago provides a complete, Microsoft supplied Microsoft Client for NetWare for Windows. This client can be installed as the default network support for Chicago, or it can coexist with the Microsoft Network client. The Microsoft Client for NetWare for Windows provides interoperability with NetWare 3.*x* and 4.*x* servers.

Chicago can also run on top of the existing Novell NetWare 3.*x* or 4.*x* clients, the NETX or VLM shells. This support is intended to help customers make the transition from their real-mode network to Chicago's fully 32-bit protected-mode implementation in smaller steps if necessary.

32-bit Microsoft Client for NetWare

The Microsoft Client for NetWare has the following key features:

- High Performance—up to 200% faster for some network operations compared to Windows 3.1 with the NetWare VLM shell installed
- Robust and reliable client support
- No conventional memory footprint
- Auto-reconnect feature
- Packet burst protocol support
- Client side caching
- Plug and Play aware
- Fully integrated into the Chicago user interface shell
- Fully interoperable with Novell NetWare 3.*x* and 4.*x* clients and servers
- Runs NetWare command line utilities
- Graphical logon to NetWare 3.*x* bindery, or 4.*x* NDS
- User-level security implemented using "pass-through" bindery or NDS
- NetWare compatible login command processor
- Point and Print support

The client is fully implemented as 32-bit virtual device driver components. The client runs in protected-mode and designed for operation in a multitasking environment, hence will be much more robust than real-mode networking components. By running in protected-mode, the drivers take no MS-DOS conventional memory space.

The Microsoft Client for NetWare has great performance characteristics. On large block transfers over the network it is up to 200% faster than Windows 3.1 and the VLM shell, in fact it's up to 200% faster than Chicago using the VLM shell. For most network operations that are a mix of reading and writing, the Microsoft Client for NetWare is between 50% up to 200% faster depending upon the mix of network I/O.

The Microsoft Client for NetWare is enabled for Plug and Play, meaning that it's possible on a portable system to hot-dock or undock a notebook computer and have the networking support properly load and unload, without hanging the system. This will also function in the same fashion for the emerging market of PCMCIA network cards. One easy way to understand how this work is to disconnect the network cable from your Chicago PC, and the system continues to function. In real-mode networks, this causes the system to hang.

Logon to Chicago is linked to either a NetWare 3.*x* bindery, or 4.*x* NDS. This logs the user onto the Chicago system and to their preferred NetWare server.

The Microsoft Client for NetWare has the ability to process NetWare login scripts. This means that if drive mappings and search drives are specified in the login script then under Chicago the same user configuration will be implemented, with no changes necessary. The Chicago login processor will parse conditional statements as well built into login scripts. However, if the login script is used to implement loading of TSRs, then the login file needs to be updated to remove these TSRs from being loaded. One key difference in login processing is the Chicago login processor operates in protected-mode, hence, loading TSRs is not possible. Rather, these TSRs should be loaded in our 16-bit driver load prior to Chicago's protected-mode operation. In some cases the TSRs loaded are backup agents, and so on. that have protected-mode equivalents built into Chicago, hence, loading these TSRs may not be necessary.

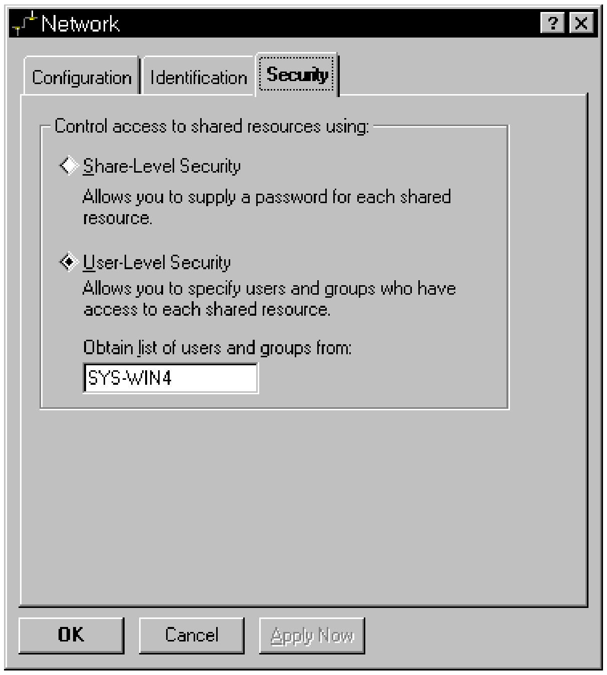
The Chicago Microsoft Client for NetWare can load and run NetWare command line utilities. It also supports the MS-DOS level NetWare APIs, and the 16-bit Windows DLLs that NetWare supplies can be run on the Microsoft Client for NetWare for Chicago.

Microsoft File and Print Sharing for NetWare

Chicago provides Peer Services for NetWare clients. The NetWare compatible Peer Services provide sharing for local files and printers on the Chicago system. During Chicago installation and via the Network Control Panel tool the option is provided to install either the NetWare Compatible Peer Services or Microsoft Network Peer Services. Chicago's Peer Services are meant to work in concert with an existing Novell NetWare server and add complementary sharing services.

For the NetWare Compatible Peer Services to be activated there must be a Novell NetWare server on the network. Without this server, sharing cannot be enabled on Chicago's NetWare Compatible Peer Services because of the "pass-through security" model.

User-level security is implemented using the NetWare servers security authority namely the Bindery or the NDS, hence "passing through" the validation of users to the NetWare server. Before sharing is enabled, a NetWare server must be specified via the Security Control Panel tool. To specify which server or Domain Controller is the designated security authority for this PC, the following dialog box is used in the network Control Panel tool:



Control access to shared resources using:

Share-Level Security [✓ User-Level Security]

Share-Level Security: Allows you to supply a password for each shared resource.

User-Level Security: Allows you to specify users and groups who have access to each shared resource.

Obtain list of users and groups from: SYS-WIN4

Figure 49. Specifying "Pass-through" From a Server Named SYS-WIN4

From the properties dialog of the PC's hard drive, Chicago gives the option to add uses to share the hard disk. If the user selects the option to add another user to this share, the following dialog box appears:

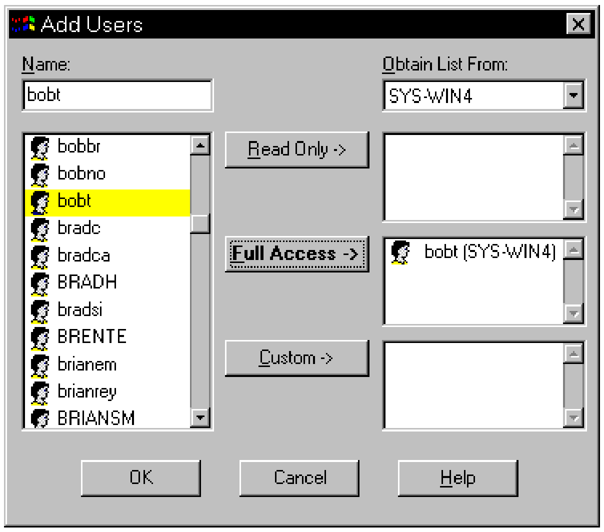


Figure 50. Configuring Access Privileges for a User Through User-Level Security

Notice that the list of users offered to add to access this directory are those from the SYS-WIN4 server's bindery.

This means two things. First, user management is all done in the namespace of the existing NetWare server. The NetWare server is administered using all the same tools that are currently in use, Chicago hasn't added another namespace to administer. Secondly, only valid user accounts and groups can be specified for sharing on Chicago NetWare Compatible Peer Services.

If the user now attempts to access a shared device on the Chicago system, the Chicago PC upon receipt of the connection request validates the user name or group membership with the NetWare server. If the name or group membership is validated, the Chicago peer services then checks if this validated name or group has been granted access rights to the shared resource and grants or denies the connection request.

Microsoft Print Server for NetWare

The Chicago Microsoft File and Print Sharing for NetWare includes a Win32-based "PSERVER" capability which can despools print jobs from NetWare queues to printers on Chicago PCs. This means that a NetWare server queue can be serviced by a printer attached to a Chicago Microsoft File and Print Sharing for NetWare system. There are certain benefits, the print queues can all be managed centrally from the NetWare server hence users print to one queue. If several Chicago Peer Services systems are on the network, each can despool from one queue increasing overall network based printer capacity. Alternatively, queues can be designated specifically for printers attached to a Chicago Microsoft File and Print Sharing for NetWare system.

NetWare 4.*x* support

Chicago's client has support for logging into and browsing the NetWare 4.*x* NetWare Directory Services (NDS). This means that Chicago's client is able to logon into the NDS tree, connect to the NetWare 4.*x* NDS tree, and navigate the tree. Accessing the tree is simple, start in the Network Neighborhood and select a NetWare 4.*x* NDS tree.

The initial contents of the Network Neighborhood are the network resource contained in the Organizational Unit (OU) context that the user. This could include NetWare servers or any other services contained within the OU. By double-clicking the server icons, the user can further browse for resources contained on this server.

NetWare API support

The Microsoft Client for NetWare includes support for both the MS-DOS APIs and Windows APIs defined by Novell. Both of the 16-bit Novell DLLs for Windows—NWNET.DLL and NWCALLS.DLL DLLs—can be run with the Microsoft Client for NetWare. This ensures compatibility of any MS-DOS or Windows applications and utilities that are NetWare-aware will run compatibly with the Microsoft Client for NetWare.

Other NetWare Interoperability

Chicago also offers these interoperability features:

- Novell command-line utilities (client and admin) are fully supported for NetWare 3.*x* currently, NetWare 4.*x* before final product shipment.
- Support for booting diskless workstations from NetWare servers
- Floppy boot capability
- Dial-up Connectivity to Novell's NetWare Connect server.

Microsoft Network Integration

Chicago includes a network client that implements support for Microsoft Network functionality. This allows Chicago to connect to Windows for Workgroups, Windows NT Advanced Server, LAN Manager and interoperate with IBM® LAN Server, DEC Pathworks™, AT&T® Starlan, and LAN Manager for Unix, as well as other SMB-compatible networks.

32-bit Microsoft Client

Key Microsoft Network client features include:

- Robust
- No conventional memory footprint
- Auto-reconnect feature
- Client side caching
- Plug and Play aware
- Fully integrated into the Chicago user interface shell
- Protocol independent
- Point-and-print for one-click printer setup

The Microsoft Network client is implemented as a collection of 32-bit, protected-mode components. The Network Provider, Redirector, and NDIS 3.1 drivers are implemented as VXDs, and hence provide great performance since the components execute in protected-mode without the overhead of switching to real-mode. The Network Provider includes the implementation of Chicago client-side caching for additional performance boost. The client has higher reliability than real-mode components, it is designed for operation in a multi-tasking environment and the components run in kernel Ring 0 context, hence they can't be touched by errant Windows applications like real-mode networks. And finally, since they run in protected-mode, they have no conventional memory footprint.

The client is enabled for key Chicago features like long filenames, links, auto-reconnect to servers, "point and print", Plug and Play, and integrated tightly into the Chicago shell via the NPI discussed previously. The client is protocol-independent, it can use IPX/SPX (the default installed protocol), TCP/IP, or NetBEUI.

The client provides full interoperability with Windows for Workgroups, Windows NT Advanced Server, LAN Manager, and LAN Manager for UNIX. It also provides compatibility with AT&T StarLAN, IBM LAN Server, 3Com® 3+Open® and 3+Share® and DEC Pathworks.

For compatibility and to help customers implement floppy boot, or better manage transition to Chicago, a real-mode client for Microsoft Networks is also included. The Microsoft real-mode components can be "unloaded" by the operating system once the protected-mode networking software is loaded.

32-bit Microsoft Network Peer Services

Chicago includes enhanced peer network services for Microsoft Networks. This is new for Chicago is user-level security. Chicago's peer services for Microsoft Networks can be linked directly to Domain based user accounts. This means that for network administrators, control over access to peer services is centrally controlled at the Domain controller. This Domain controller can be either a Windows NT Advanced Server or a LAN Manager domain controller.

User-level security begins with sharing a device on a Chicago system. The list of users that appears in the sharing dialog box are provided by the Domain controller, hence it's only possible to share the device to validated Domain users. The share is established and user logons are now specified for access rights. When a user requests access to a shared Chicago resource, the Chicago peer services PC checks for the users logon name against the domain controller. If this is a valid user logon, the Chicago peer services then checks if this user has access privileges for this resource. If the user logon has access privileges, then the user connection is established.

Like Windows for Workgroups, Chicago includes share-level peer services. This level of security associates a password with a share of a disk directory or printer. Share-level security can be implemented in a Chicago-only network, or on a network with other Microsoft Networks compatible servers.

Chicago peer services are remotely administerable, via the NetWatcher tool a network manager can monitor connections to any resource on any Chicago peer services PC on the network. The network manager can then disconnect any users, and remotely change access rights for this user on the specified Chicago peer services PC. By default, remote administration is limited to user accounts with "administrator" privilege.

Chicago peer services also includes auditing of events like user logon, attempted logon, server start/stop date and time stamps. This event log is also remotely accessible using the NetWatcher tool. A network manager can remotely access the log, turn it on or off, change events to monitor, or restart the log as needed.

Network Compatibility

Chicago includes built-in support for Microsoft Networking and Novell NetWare. However, Chicago's Setup can correctly install and configure itself for a variety of existing real-mode networks, including, but not limited to the following:

- 3Com: 3+Open, 3+Share
- Artisoft LANtastic®
- Banyan VINES®
- Beame and Whiteside: B&W-NFS
- DEC PATHWORKS
- IBM: LAN Server and LAN Program and PC LAN Program
- Microsoft LAN Manager, MS Net
- Novell NetWare
- SunSelect PC-NFS
- TCS 10net

Protocol Support

Chicago protocols are implemented as 32-bit protected-mode components. Chicago can support multiple protocols simultaneously. Protocol stacks can be shared among the networks that are installed. As an example, a single TCP/IP protocol stack can service both the needs of the Windows Networks client and the Microsoft Client for NetWare.

All three protocols included with Chicago (IPX/SPX, TCP/IP, and NetBEUI) are Plug and Play enabled. This means that if the network is unavailable either due to undocking a notebook PC, or removal of a PCMCIA network card, the Chicago system continues to run. The protocol stacks will unload themselves after having notified any dependent applications that they will be unloaded from the system. Additionally, this also means protocols can automatically be loaded. For example, if a mobile PC user goes from network attached to an infrared (IR) line of sight network, the TCP/IP protocol can be unloaded and the appropriate IR protocol loaded, automatically.

IPX/SPX

The IPX/SPX stack is the new default protocol for Chicago and is compatible with the Novell NetWare IPX/SPX implementation. This protocol stack can be used to communicate to either a NetWare server, or a Daytona Windows NT Advanced Server. This protocol is routable, and will run compatibly on most network infrastructure (such as bridges, routers, and so on.) that are designed for IPX/SPX routing. Chicago's IPX/SPX protocol includes support for "packet burst" which can offer improved network performance.

One enhancement made to the Microsoft IPX/SPX implementation is Windows Sockets programming interface support. The Windows Sockets interface is supported using IPX/SPX as the protocol. Hence, any WinSock applications can run on top of IPX/SPX with Chicago. Support is provided for only Win32 WinSock applications.

The Chicago IPX/SPX implementation also has support for the NetBIOS programming interface.

TCP/IP

For connectivity to the Internet, or for many corporations implementation of an industry standard network protocol, TCP/IP is becoming widely accepted. Chicago's implementation of TCP/IP is a 32-bit VxD, is high performance and consumes no conventional memory.

Chicago includes a full TCP/IP implementation that includes several of the more commonly used command line utilities, which include telnet, ftp, arp, ping, route, netstat, nbstat, ipconfig, tftp, rexec, rcp, rsh, and traceroute.

The Chicago TCP/IP protocol support includes the Windows Sockets programming interface, and includes a WinSock DLL. Support is provided for both 16-bit WinSock for compatibility with existing WinSock applications, and 32-bit WinSock for Win32 WinSock applications.

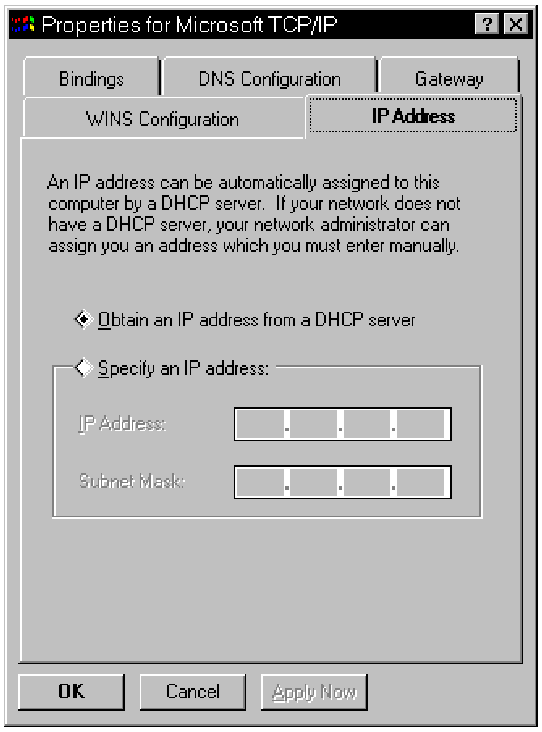
A NetBIOS programming interface support is also supplied with the TCP/IP support.

DHCP Support

In an effort to make implementation of the TCP/IP protocol more manageable, Microsoft, working with other industry leaders have created a bootp backward-compatible mechanism for automatic allocation of IP addresses. The Dynamic Host Configuration Protocol (DHCP) runs from a Windows NT DHCP server, and it allocates a network manager to centrally establish a range of IP addresses per subnet automatically to any Chicago TCP/IP client requesting and address. It also allows the network manager to centrally establish a "lease time" or how long the allocated IP address is to remain valid. Unlike bootp, the address allocation is dynamic, not pre-configured. In this fashion it's possible to move from subnet to subnet and always have a valid IP address mask. Chicago includes a ipconfig utility that allows a user or administrator to quickly examine the IP address allocated, lease time and other useful data about the DHCP allocation, as shown below.

```
Windows IP Configuration Version 0.1
     Host Name . . . . . . . :
     DNS Servers . . . . . . :
     DNS Lookup Order. . . . :
     Node Type . . . . . . . : Mixed
     NetBIOS Scope ID. . . . :
     IP Routing Enabled. . . : No
     WINS Proxy Enabled. . . : No
     WINS Resolution For Windows Sockets Applications Enabled  : No
     DNS Resolution For Windows Networking Applications Enabled : No

Adapter Address 00-AA-00-18-B0-C4:
     DHCP Enabled. . . . . . : Yes
     IP Address. . . . . . . : 11.105.43.177
     Subnet Mask . . . . . . : 255.255.0.0
     Default Gateway . . . . : 11.105.0.1
     DHCP Server . . . . . . : 11.105.43.157
     Primary WINS Server . . : 11.101.13.53
     Secondary WINS Server . : 11.101.12.198
     Lease Obtained. . . . . : Tue 10th. May 1994  6:44:40 am
     Lease Expires . . . . . : Wed 11th. May 1994  6:44:40 am
```

DHCP support can be specified at install time, or enabled via the Network Control Panel tool. If the user prefers, a "hand-entered" IP address can be used and DHCP support can be disabled.


Properties for Microsoft TCP/IP

Bindings | DNS Configuration | Gateway | WINS Configuration | IP Address

An IP address can be automatically assigned to this computer by a DHCP server. If your network does not have a DHCP server, your network administrator can assign you an address which you must enter manually.

[✓ Obtain an IP address from a DHCP server]

Specify an IP address:

IP Address: . . .

Subnet Mask: . . .

Figure 51 Properties for Microsoft TCP/IP Showing DHCP Configuration

WINS Support

The Chicago TCP/IP protocol stack lets the user choose to install support for either the Windows NT Windows Internet Naming Service (WINS) or the OSF DCE Domain Naming Service (DNS). The naming services provides name resolution by binding the node name and the currently allocated IP address. This provides for correct addressing of any requests for resources from a node anywhere on the network, thus minimizing the amount of network traffic to locate the node on the network. Chicago supports a single DNS server and up to two WINS servers.

NetBEUI

Chicago provides a NetBEUI protocol stack that's compatible with existing networks that use NetBEUI. This provides compatibility with Windows for Workgroups, Windows NT Advanced Server, LAN Manager, and other networks. A NetBIOS programming interface is also supported.

Network Interprocess Communications Interfaces

Chicago includes support for a variety of distributed computing programming interfaces, these include:

- Client-side named pipes
- Mail slots
- OSF DCE compliant Remote Procedure Call (RPC)
- Network DDE
- Windows Sockets Interface

Long Filename Support

Chicago's network clients include long filename support. If the network server that the Chicago system is connected to supports long filenames, then filenames on the server will include identical support to Chicago's local long filename support. On some servers the length of filenames and restricted characters may differ from Chicago's. This means that it's possible to have long filename support on both the Windows NT Advanced Server and NetWare servers if the servers are properly configured.

Network Printing

Chicago includes a number of enhancements designed to make printing easier over the network including:

- **Point and Print.** Automatic installation of a printer driver when connecting to a printer attached to a Novell NetWare, Windows NT Advanced Server, or Chicago print server. Chicago printer drivers can be located on Novell NetWare servers and Windows NT Advanced Server servers and automatically installed by the Chicago clients.
- **Microsoft Print Server for Netware.** Chicago peer services can "despool" print jobs from Novell NetWare print queues. This is compatible with NetWare's Pserver functionality.
- **Deferred Printing.** When the Chicago PC is disconnected from the network, print jobs are deferred until a later date when the PC is once again attached to the network. Print jobs that have been deferred will automatically be started when the PC is reconnected to the network.
- **Remote Printing Management.** Print jobs can be held, canceled, or restarted remotely. In addition, on systems that have ECP ports, more information about the print job status can be returned concerning paper tray status, paper jams, or other error conditions.

Network Security

Chicago implements a full user logon. The first thing most users will encounter after booting their Chicago system is a logon dialog box. This dialog box varies depending on the type of network that they are logging into. For example, the Windows NT Advanced Server logon dialog may prompt the user for a username, password, and domain name. The Novell NetWare 4.*x* logon may prompt the user for a username, password, and NDS OU name. Once the username and password pair have been validated against the network server's user authentication, the user is allowed in to the Chicago user interface.

If the user fails to logon, the network manager can configure the Chicago system to allow entry into the Chicago user interface, albeit with no network access. This is the default configuration. Additionally, administrators can specify guest accounts that have more limited network access as an alternative solution to this problem.

However, because Chicago has a user logon, it's not to be construed as a mechanism to fully secure the PC. The PC is still vulnerable to a boot floppy and thus all data stored on the hard disk is available. The underlying file system in Chicago is the MS-DOS FAT file system, hence has no encryption or other security mechanisms built-in.

Instead, Chicago's focus is to provide that network resources are secured using the same security mechanisms in place today via the network server on the corporate network. The Chicago username and password can be configured to be the same as those used by the network server, and can thus control network access, user-level security for access to shared resources on this PC, control of the various Chicago agents, as well as limiting who has remote administration authority on this Chicago system.

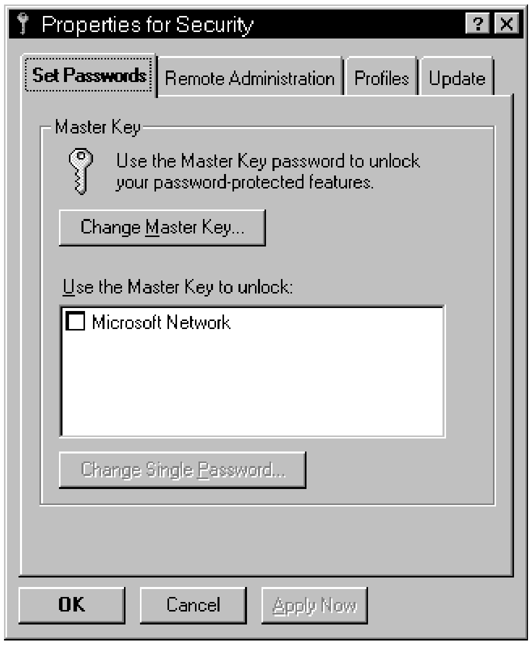
In this fashion, Chicago leverages the existing investment in network servers, management tools, utilities and infrastructure. Network managers can manage user accounts centrally on the server, just as they do today. They can also use the same tools that they do today for managing the user accounts.

Master Key: Unified Logon

Chicago's Master Key can provide a unified logon for all system components requiring password authentication services, as well as any applications that choose to use the Master Key services. For example, protected spreadsheets, or database access may use the Master Key services.

Master Key associates the username and password supplied at Chicago logon to other authentication conscious programs or system components. However, it's also possible to maintain separate password, in essence for higher security a network manager may choose to associate other passwords with vital corporate data access or other sensitive network services.

The figure below shows the Master Key dialog box from the Control Panel:



Use the Master Key password to unlock your password-protected features.

Use the Master Key to unlock:

☐ Microsoft Network

Figure 52. Properties for Security, Showing Master Key Password Settings

Note the Master Key provides a mechanism to individually manage components that choose to use the unified password cache. On a service by service, or application by application basis, Chicago can be configured to use the Chicago logon for authentication. This makes it possible for a user to achieve a "single logon" for access to all resources on the Chicago system, as well as the network using Chicago's Master Key. One example of how the Master Key service is used within Chicago is providing a single logon to both the network and the Chicago Mail Client (code named Capone). Once a user has logged onto their Chicago PC, the password they entered to logon to Chicago will also automatically log them onto email. This finally provides a solution for the password proliferation problem that confounds many users today.

User-level Security

Chicago uses the logon process for user-level security to implement control for a variety of services beyond network resource access, including many services running on a Chicago system that are remotely accessible, namely:

- File and printer sharing
- Remote network access gateway control
- Backup agent
- Network and system management

Pass-through Security

Pass-through security is implemented in Chicago as the mechanism to enable user-level security. Pass-through quite literally means that Chicago passes authentication requests through to a Windows NT or NetWare server. Chicago does not implement its own unique user-level security mechanism, rather it uses the services of an existing server on the network.

File and Print Sharing

For file and print sharing using Chicago peer services, enabling pass-through security is a two step process. First, using the Control Panel user-level security must be enabled. The second step is to share a device, and specify users with access privileges. By clicking the secondary mouse button on the drive C icon in "My Computer," a properties sheet is revealed that allows sharing on the second tab. This property sheet shows what shares exist, and which users have access.

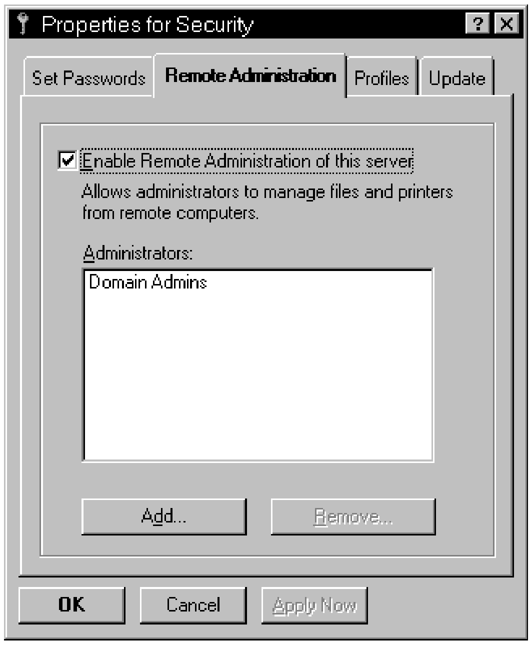
The user names returned in this property sheet are returned from either the Windows NT Advanced Server domain or NetWare bindery or NDS.

Remote Administration

Remote administration of the Chicago PC specifies users or groups that have authority to manage the Chicago system. This includes:

- Remote network access gateway control
- Backup agent
- Remote access to the Registry
- Remote NetWatcher access
- Remote system performance monitoring

Remote administration is controlled via the Network Security tool in Control Panel. The figure below shows remote administration enabled:



[✓] Enable Remote Administration of this server

Allows administrators to manage files and printers from remote computers.

Administrators:

Domain Admins

Figure 53. Properties for Security, Showing Remote Administration Settings

In this case, remote administration is limited to the network manager group of "Domain Admins." Any user that is a member of the supervisor group can remotely administer this Chicago system. It is also possible to specify user names in addition to groups for remote administration capabilities. For example, sophisticated users may be given remote admin access to their systems.

Remote Network Access Gateway

Chicago includes a single line, dial in gateway that allows a Chicago PC with Peer Services enabled to serve as a gateway to the network. The RNA Gateway supports the same protocols as the RNA client, namely:

- TCP/IP using the Point to Point Protocol (PPP)
- IPX/SPX via PPP
- NetBEUI

The RNA Gateway also implements pass through security, so only authenticated users are allowed to logon to the Gateway services. Once connected to the Gateway, RNA clients can access any network resource that they have privileges to use. This includes network server resources, or peer services.

Systems Management

Chicago is the first version of Windows expressly designed for manageability. The design ensures that management of the Chicago PC is accessible both locally, and remotely via a privileged network manager. Network security is used to determine administrator privileged accounts using pass through security. Chicago also provides for a logical separation of the user of the PC, from the underlying configuration of the PC. This means that the PC and the user configurations and privileges can be managed independently. It also means that if a network manager chooses, a user can be enabled to "rove" on the network, that is logon from virtually any PC on the network and operate in their desktop with the correct settings and network privileges. Additionally it means that a single PC can be shared by multiple users, each with a different desktop configuration and differing network privileges.

Given the proliferation of PCs connected to the corporate network, it's key that the Chicago PC also participates in any network wide management schemes. Chicago is designed to meet these various network management criteria by providing built-in support for several of the key network management standards. With this infrastructure built into Chicago, network management applications are enabled that will provide tools for the network manager to keep the PCs and networks running more efficiently and cost effectively.

Key to the Chicago management implementation is that the management interfaces are open. Where a standard exists, Chicago implements an enabling technology to embrace the standard. For example, supplying an SNMP agent to enable remote management of Chicago PCs via any number of third party SNMP consoles. Where no standard exists, the management interfaces are documented in the Win32 API set. It is Microsoft's expectation that management software will be available for Chicago from a wide range of vendors.

The list below outlines the key components of Chicago's management infrastructure:

- Chicago's Registry
- Registry Editor
- User Profiles—user component of the Registry
- Hardware Profile— system component of the Registry
- System Policies— network and system policy component of the Registry
- System Policies Editor
- Remote Administration Security—remote admin authentication scheme
- Remote Procedure Call—mechanism used to remotely admin Chicago
- NetWatcher
- System Monitor—performance monitor
- SNMP Agent
- DMI Agent
- Tape Backup Agents—ARCServe, Arcada "MTF"

The discussion of Chicago's management infrastructure is organized as follows:

- The Chicago Registry
- User Management
- System Management
- Network Management

The Chicago Registry

The Registry is the central repository in which Chicago stores the whole of its configuration data. The Chicago system configuration, the PC hardware configuration, Win32 applications, and user preferences are all stored in the Registry. For example, any Chicago PC hardware configuration change that's made via a Plug and Play device is immediately reflected in a configuration change in the Registry. Because of these characteristics, the Registry serves as the foundation for Chicago user, system and network management.

The Registry essentially replaces the various MS-DOS and Windows 3.11 configuration files, including AUTOEXEC.BAT, CONFIG.SYS, WIN.INI, SYSTEM.INI and the other applications .INI files. However, for compatibility purposes, instances of CONFIG.SYS, WIN.INI and SYSTEM.INI files may exist on a Chicago PC for backward compatibility with either 16-bit device drivers, or 16-bit applications that must run on Chicago. For example, we expect that 16-bit applications will continue to create and write to their own various .INI files.

The Registry concept is built upon the Registry concept first implemented in Windows NT. The Registry is the single configuration datastore that's built directly into the operating system. The Registry is logically one datastore, but physically it consists of three different files to allow maximum network configuration flexibility. Chicago uses the Registry to store information in three major categories:

- User specific information, these are user profiles contained in the file USER.DAT.

- Hardware or computer-specific settings are contained in the file SYSTEM.DAT.

- System policies are designed to provide an override to any settings contained in the above two components of the Registry. System Policies may contain additional data specific to the network or corporate environment as established by the network manager. This is contained in the file POLICIES.DAT.

Together these three components comprise the Registry. By breaking the Registry into these three logical components, Chicago gains a number of interesting benefits:

- The Registry components can be located in physically different locations. For example, the SYSTEM.DAT component and Chicago's other system files may be located on the PC's hard disk. The USER.DAT portion of the Registry may be located in the user's login directory on a network server. In this configuration, user's are able to logon to various PCs on the network and still have their unique network privileges and desktop configuration, thus allowing the “roving user” network configuration for Chicago.

- All three of the Registry files and the rest of Chicago’s system files can be installed on a network server. This configuration enables Chicago to be run on diskless remote initial program load (RIPL) workstation, or from a floppy disk boot configuration. In this scenario, it’s also possible to configure Chicago to page to a local hard disk if desired, but still load all it’s system files from a server.

- On a single Chicago PC, multiple users can share the system. Each will have a separate user logon name, and separate user profiles. Hence each use will have their own privileges set, and own desktop configurations. In this case, the Registry and all Chicago system files are installed on the local hard disk.

- Network managers can administer an entire networks users privileges via a single file. By having a global POLICIES.DAT file, effectively all Chicago PCs can have policies set by this one file. Or, these policies can be established on a server basis, or if needed, on a per-user basis. In this fashion a network manager can enforce a “common desktop configuration” for each end user type and have this managed centrally. For example, a data entry desktop can be configured to allow only two applications available to run, the data entry application and email as an example. Additionally this desktop can be configured to not allow any other programs to be run. Finally, the network manager can enforce that the desktop configuration cannot be modified by the end-user. However, this same Chicago PC can fully participate in the network and be fully configurable if a different user with more network privileges logs onto the same PC.

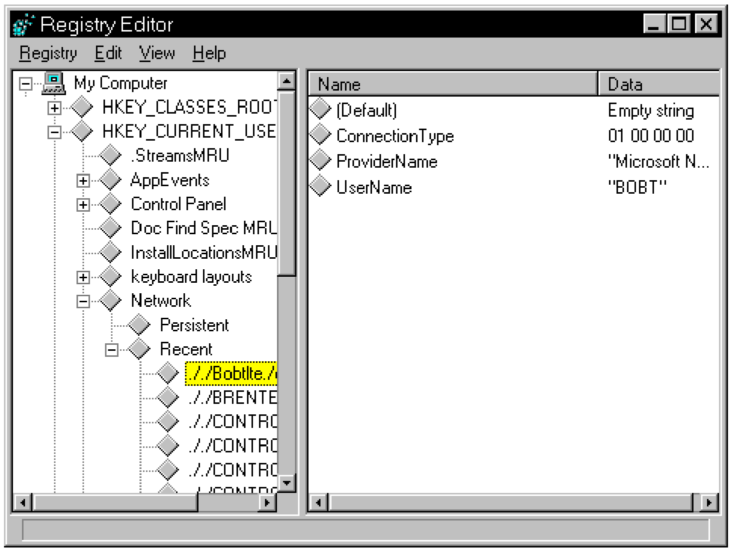
- Separate privileges can be assigned to users and to a PC. For example, it’s possible to have set no sharing (no peer services) on a Chicago PC, and have a user logged on the system that has sharing privilege. In this instance, the sharing is disabled, since the resources on the PC have been set as unshareable. This feature is useful if certain PCs contain sensitive data that should not be “shareable” to the corporate network.

The Registry contains ordered pairs of “keys” and their associated “values.” Both keys and values are manipulated via the Win32 Registry APIs. An example, a key in the Registry could be “NETWORK_CARD_TYPE”. The associated value in this example could be “TOKEN_RING”. In this example, this means that the network topology is configured for token-ring operation.

Additionally, there exist a special category of keys known as *dynamic keys*. Dynamic keys are either pointers to a memory location, or a call back function. These addresses are registered by a device driver or a Chicago subsystem that would like to register a dynamic data type in the Registry. Typically this data includes counters, or in the case of network cards the dynamic keys represent things like data transfer rates, number of framing errors, packets dropped, and so on. In general, the characteristic of dynamic keys is that its data is being updated frequently, and because of this, is not well suited for storage in the disk based Registry. The dynamic keys exist in memory, and can thus be quickly updated, and quickly accessed. This data can then be accessed by the Chicago system performance tools, which call the Registry for the data that they are monitoring. Dynamic keys are a Registry enhancement new for Chicago.

Arbitrary keys and values can be created programatically, or using the Registry Editor (regedit) tool. The API for managing the Registry are the Win32 Registry API. These APIs can be remotely invoked via the Microsoft RPC (DCE-compliant) support built into Chicago. Chicago includes both the client and server portions of our RPC, making Chicago's Registry remotely manageable from another Chicago PC. In this scenario, the network manager's system is the RPC client, it accesses the Registry APIs on the target Chicago PC via the RPC server running on the target machine. This RPC access to the Chicago Registry is secure, network managers can limit access to either named privileged user accounts, or a group of network managers.

The Registry is also editable using the Registry Editor utility, as pictured below:



Name	Data
(Default)	Empty string
ConnectionType	01 00 00 00
ProviderName	"Microsoft N...
UserName	"BOBT"

Figure 54. Network Settings are Stored in the Registry, and Can Be Accessed Remotely

Note in the photo that the Registry consists of various parallel "trees." The RegEdit utility is built upon the RPC support, and can edit the local Chicago Registry, as well as editing the Registry on a remote Chicago PC. The RegEdit tool while very powerful, if fairly rudimentary in it's design. The utility is designed to be used by knowledgeable PC and network support staff, or power users. For most end users, Registry entries are modified through either the Control Panel, application settings, or via Plug and Play. That is, typically an end user should not be confronted with a scenario where they must use the Registry Editor tool.

User Management

Chicago is the first version of Windows to implement functionality for management of user specific configurations and user specific privileges. User management under Chicago is most evident with the introduction of a user logon dialog that minimally prompts the user for their logon name and password each time they reboot the Chicago PC. This logon dialog captures the username and password that can trigger Chicago to dramatically reconfigure the desktop configuration and as needed, limit access to either network resources or sharing capabilities from this Chicago PC. Chicago can also pass the user logon and password to registered applications and network services that use Chicago's logon as the "Master Key" to enable the user access to these applications and services.

Chicago's User Management capabilities are built upon the following components:

- User Profiles
- System Policies
- Server Based Security

User Profiles

In Windows 3.11 settings unique to a user were located in many disparate locations; AUTOEXEC.BAT, CONFIG.SYS, WIN.INI, SYSTEM.INI and numerous application specific INI files. For example, this data was often intertwined with Windows internal configuration data, thus providing good user management using Windows 3.11 was very difficult to achieve. For example, the simple task of allowing multiple users to use a single PC was not achievable with Windows 3.11 "out of the box." Managing multiple user configurations on a network was even more difficult. Many companies out of necessity wrote their own user management tools, or used third-party tools to help mange multiple users on the network. Very often this user namespace did not leverage the existing namespace on the corporate network resident on the network servers. In some cases, the user management software was implemented as a replacement Windows shell, with varying degrees of compatibility with the existing Windows applications and the underlying network client software. All these tools and products attempted to retroactively address Windows 3.11's lack of user management capabilities.

Chicago's user management is integral to the system, it is implemented in a feature known as User Profiles. User Profiles are part of the Registry, they contain system, application and network data that are unique to the individual users of a Chicago PC. These characteristics can be set by the user, by the network manager or by the PC helpdesk staff. In contrast to Windows 3.11, Chicago's User Profiles are contained within a single file named USER.DAT. By keeping all user specific data in one file, Chicago can provide a means to manage the user of the PC separately from the configuration of the Chicago OS and of the PC hardware. This separation also allows the user information to be located in a physically different location than that of the system configuration. It also allows the User Profiles to be updated separately from the rest of the Registry. All settings contained within a User Profile are administerable locally or remotely from another Chicago PC, Chicago enables centralized user management. The network manager can user Chicago's built-in Registry Editor, or a variety of third party tools that will be available to automate management of Chicago's User Profiles.

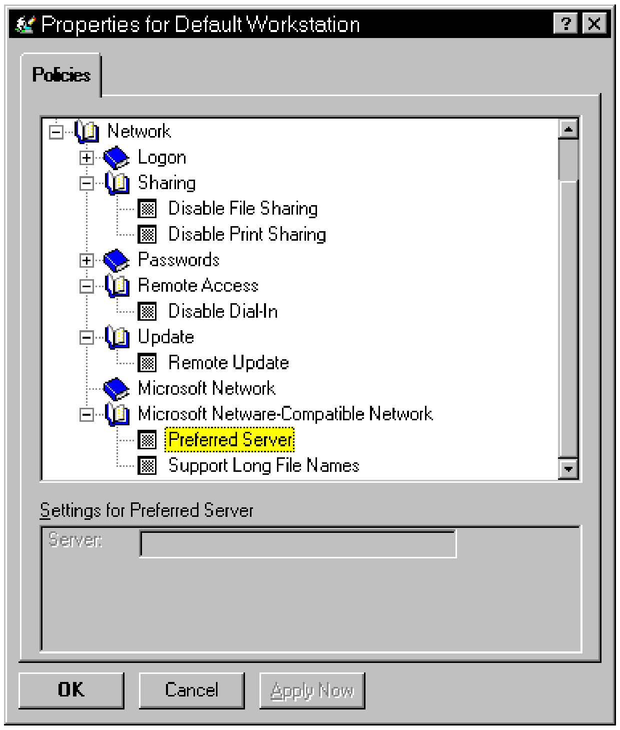
In Chicago, settings contained in User Profiles include:

- **Chicago Settings.** Desktop layout, background, font selection, colors, shortcuts, display resolution, and so on.
- **Network Settings.** Network connections, workgroup, preferred server, shared resources, and so on.
- **Application Settings.** Menu and toolbar configurations, fonts, window configuration preferences, and so on.

Finally, User Profiles can effectively be disabled if there is a single user of the Chicago PC. In this case the user can disable "each user gets a new desktop" option in the Control Panel.

System Policies

In conjunction with User Profiles and the System Settings components of the Registry, System Policies are the final piece of the Registry. Like the other two Registry components, the System Policies consist of pairs of keys and values. Unlike the other two Registry components, System Policies are designed to override any settings that may exist in User Profiles or System Profiles. System Policies are not necessary to enable a Chicago system to boot. System Policies are loaded last, and are typically downloaded from a network server. Chicago provides a mechanism to allow the network manager to define a network location to find the System Policies file and download to this PC. System Policies are designed to give the network or PC manager the ability to customize control over Chicago for users of differing capabilities or network privilege level. These capabilities include controls of the user interface, network capabilities, desktop configuration, sharing capabilities, and so on.


Properties for Default Workstation

Policies

```
Network
    Logon
    Sharing
        Disable File Sharing
        Disable Print Sharing
    Passwords
    Remote Access
        Disable Dial-In
    Update
        Remote Update
    Microsoft Network
    Microsoft Netware-Compatible Network
        Preferred Server
        Support Long File Names
```

Settings for Preferred Server

Server:

Figure 55. Policy Properties for a Default Workstation

System Policies may be used to define a “default” setting for either the User Profile or System Settings. Default settings for both a default user and a default computer may solve the problem of preconfigured PCs for network managers. New PC hardware comes pre-installed with Windows and in some cases network hardware and software necessary to connect to the corporate network out of the box. Many network managers today have network-wide standard Windows 3.11 that normally are pre-configured by hand on each PC before allowing it on the corporate network. However, if PCs are directly fulfilled to end-users, as is often the case, the network manager will not have the opportunity to install the network wide standard configuration on this PC. However, the default System Policies may solve this problem. For example, assuming that the standard Windows configuration consists of a number of corporate standard applications and a standard set of network privileges (for example, servers they are allowed to connect to) then the default settings will “enforce” these standards the first time the PC is connected to a network server. Assuming that the user logs on with their existing user logon name, then the network privileges that they’ll see are exactly those that they are entitled to today. In this case, the assumption is that the network manager had pre-configured a user based set of system policies.

The range of desktop control offered by System Policies is fairly comprehensive. The network manager can define a desktop for a user, then make the lock down this desktop configuration. This is accomplished by turning on the attribute that the desktop in unmodifiable by the user. Additionally, the network manager can further insure that the user only has access to the applications that they've installed by disallowing the user to run any additional programs. This means that the user cannot run programs from the command line or from the UI browsers, thus preventing them from installing additional software. Some other examples include disabling elements of the Control Panel for users that may have the habit of reconfiguring their PCs and are thus are perennially "helpdesk intensive." As noted before, standard network connections, enabling and disabling of peer sharing capabilities and things like password aging, can all be implemented using the System Policies feature.

Registry Tools

Chicago's primary user management tools are the Registry Editing and System Policies Editing tools. For most other user administration, network managers will use the user accounts tools on their PC servers that they already use today.

Registry Editing Tool

The Registry Editing Tool allows the network manager to directly read and write values that are contained in the User Profiles and System Settings portions of the Registry. Using this tool, it's possible to read current settings, modify them, create new keys and values or delete current keys and values in the Registry. The Registry Editing tool is able to edit remote Registry's, using the RPC enabled Win32 Registry APIs built-in to Chicago.

In the case of the User Profile residing on a network server, the network manager simply connects to the network server, and opens the file using normal file I/O. In this case, there is no RPC connection between the Chicago client and the network server.

Admin Config Tool

The Admin Config Tool generates the System Policies file, POLICIES.DAT. This tools allows the network manager to specify specific network policy or user configurations for Chicago. The tool is extensible by third parties, the ADF format is a text file that can be extended by other network tool vendors, or network managers as needed. This tool works via local file I/O, and is not RPC enabled. Since the System Policies file is to located centrally on a network server, typically one copy is needed per server. Hence, all the network manager needs to do is connect to the network server, and edit the System Policies file.

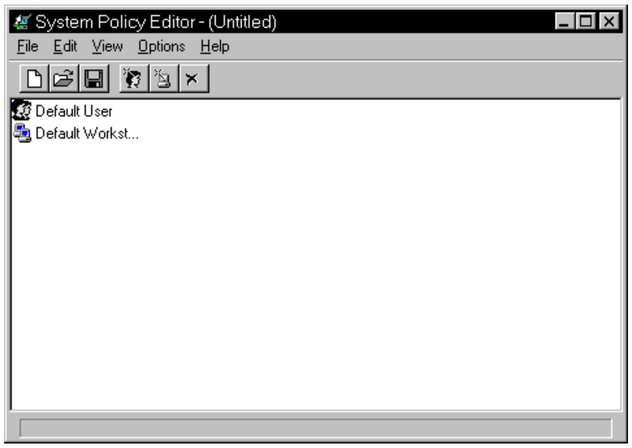

System Policy Editor - (Untitled)
File Edit View Options Help
Default User
Default Workst...

Figure 56. The System Policy Editor in Chicago Enables Administrators to Define Policies on a Per-User Basis

Role of the Server in Systems Management

In user management, the server plays a central role. All user "namespace" management is done on the network server. This means for user logon authentication, and pass through security the native user-level security mechanism built into the network server is used by Chicago. Chicago has no built-in user-level security mechanism of it's own. As a consequence, the network managers use the familiar server administration tools to manage user accounts for Chicago.

The second role of the server in Chicago user management is to contain copies of User Profiles and System Policies. Typically, User Profiles are contained in user directories, and should be read/write enabled for the user. As changes are made to the local Chicago copy of the User Profiles, they are updating the image that resides on the server. System Policies should be located in a directory that is accessible to all user logons, and should be made read only for users. This ensures that only network managers will have the rights to modify the network wide policies that the System Policies file may define.

System Management

Chicago Systems have been designed to be managed well, both locally and remotely, using the Registry's remote capabilities. The Registry enables a network manager remote management of Chicago's system software settings, including settings used by device drivers. For example, it would be possible for a network manager to remotely change the network frame type in use on all the PCs under their oversight. Currently this is done in many cases by hand directly editing NET.CFG or PROTOCOL.INI files.

Plug and Play makes Chicago PCs much more manageable in hardware configuration. It also helps address one of the paramount problems facing helpdesk staff and users, that of proper hardware configuration. One of the more complex hardware/software configuration problems revolves around the use of notebook PC docking stations. Typically this means that the user has a “boot configuration” manager in use to help manage the different devices that need to be installed while docked, or while remote. Creating these configurations is very time consuming, and often must be done for each system setup due to conflicts in other device drivers that may be installed. Plug and Play automates this docking problem, as well as PCMCIA cards and helps with link management when moving from fast links, to slower asynchronous links. The Chicago system detects these events, docking/undocking, PCMCIA card insertion/removal or moving from a fast media to a slow media. It then appropriately loads/unloads device drivers and configures them automatically. Finally, Chicago notifies applications that the device is either available, or now unavailable.

Chicago Tools

Chicago includes a variety of tools that allow a user or network manager to configure the hardware and software on a Chicago PC. These include the following:

- **Control Panel.** Traditionally, the only interface available to directly modify the configuration of hardware and software settings in Windows. The Control Panel in Chicago like its Windows predecessor is extensible, and provides the best local mechanism for managing all system settings. Most key system settings are accessible via the Control Panel. In Chicago, all network settings have been consolidated into a single network Control Panel tool, rather than split between several discrete applications as in prior versions of Windows.
- **Context Menus and Property Sheets.** Context menus and property sheets offer a number of actions that can be directly applied to system objects. They are invoked via a right mouse button click. For example, the properties menu item in the context menu for a directory with sharing enabled allows the user to invoke sharing of the directory. Another example, the properties item of a server tells what type of server this is, namely a NetWare server, a Windows NT Advanced Server server, or a Chicago system.
- Plug and Play. The current hardware configuration for the system is accessible via the Control Panel. Within the system tool, all hardware device nodes in the hardware tree are shown, with current configuration settings. These settings are updated dynamically whenever a device’s configuration changes, or if the device is inserted or removed.
- **Registry Editor.** For network managers or PC helpdesk, the Registry Editing tool allows remote viewing and editing of the full Chicago Registry. Data

contained in the Registry is represented in its hierarchical tree structure as pairs of keys and values.

- **System Policies Editor.** System capabilities can also be enabled or disable System Policies Editor. For example, sharing can be disabled on a machine basis, or local Control Panel usage can be disabled for non-privileged users.
- **DMI Agent.** Remote desktop management will be possible via the DMI agent for Chicago. This includes both hardware and software inventory, and the ability to make remote changes to the system.

Performance Monitoring

Chicago includes an enhanced performance monitoring utility. This gives network managers and PC helpdesk the ability to more quickly troubleshoot performance problems caused by invalid configuration or some other conflict. The System Monitor is the replacement for Windows for Workgroups' WinMeter. It provides more detailed information about the system's I/O performance, which includes file I/O performance and network I/O performance. Data is gathered on an FSD basis, which means it's possible to gather information from the FAT file system and any number of network redirectors that may be loaded. The interfaces to the System Monitor are open, and are extensible by third parties.

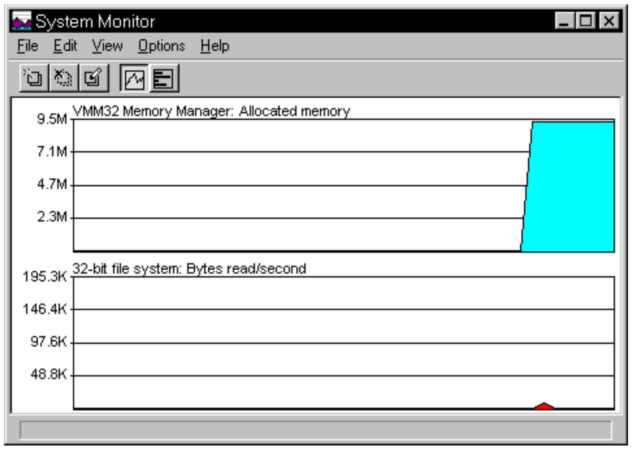

System Monitor
File Edit View Options Help
VMM32 Memory Manager: Allocated memory
9.5M
7.1M
4.7M
2.3M
32-bit file system: Bytes read/second
195.3K
146.4K
97.6K
48.8K

Figure 57. The System Monitor Tool in Chicago Allows Local and Remote Monitoring of System Performance

However, for network managers the key feature of System Monitor is its ability to monitor a remotes system. This capability is built upon remote Registry access, since performance data is registered with the system using "dynamic keys" contained within the Chicago Registry. For example, if a PC helpdesk person is attempting to troubleshoot a "slow PC," they can discover remotely that the NIC has an unusually high number of dropped frames. They can then move on to use the Registry Editing tool to see how the network card is configured.

Network Management

Chicago has included a number of features to facilitate the use of a variety of network management tools. Many of these tools by necessity require support in the client to enable their operation. In some cases a formal industry standard exists, and others, a de facto standard has emerged. In either case, Chicago enables some of the key network management tools by including the necessary "agent" software built-in to the client operating system.

Server-based Backup

Chicago includes agents for remote backup of the Chicago system by a server based backup system. Agents included with Chicago are:

- Cheyenne ARCServe agent for backup to NetWare and Windows NT Advanced Server servers
- Arcada Backup agent for backup to Windows NT Advanced Server and NetWare servers.

By including these agents, it's now possible to include Chicago systems in an scheduled automatic remote backup scheme managed centrally via the server based backup system.

Both backup agents include a number of enhancements for Chicago. For example, both agents will include the ability to backup and restore long filenames, even if the native tape format does not include a mechanism for storing long filenames. In this case, the agent includes special logic to facilitate saving and restoring the long filenames. Both agents also have been enhanced to backup and restore the Chicago Registry.

Another enhancement for Chicago, is securing operation of the backup agent by the user-level security. By default, remote administration of the Chicago PC is enabled only for "supervisor" privileged accounts. This means that only network managers or PC helpdesk staff have the ability to remotely backup Chicago systems. For example, it's key that only authorized personnel backup the hard disk of the CEO's system, or of the corporate controller's PC.

Network Management Tools

There is an emerging category of tools in the market that all claim to be network management tools. Many of the tools were actually designed to solve a specific problem, and have been extended to become more general purpose network management tools.

SNMP Support

Simple Network Management Protocol (SNMP) consoles are a good example of this trend, now being enhanced to monitor components of desktop systems, as well as server applications like database servers. Chicago includes an SNMP agent to support the use of an SNMP console to manage Chicago PCs. The SNMP support in Chicago includes:

- SNMP Agent
- Extensible MIB handler interface
- MIB II support via TCP/IP

Chicago's SNMP agent is extensible via it's MIB handler interface. This enables third parties to include instrumentation of their software or hardware components and allow remote management via the SNMP console.

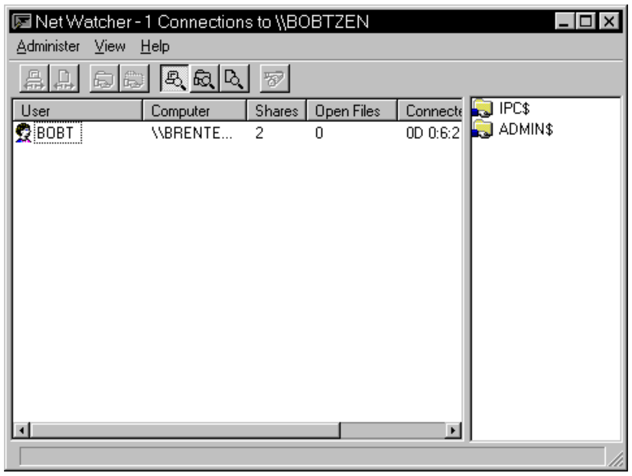
Since many corporations are beginning a migration to TCP/IP as a standard protocol, Chicago's TCP/IP stack has been instrumented for SNMP remote management. The MIB II supports the Internet Engineering Task Force (IEFT) Request for Comment (RFC) for the TCP/IP MIB definition. This can offer the network manager the capability to centrally monitor the performance of TCP/IP on the network from a central console.

DMI Support

Chicago will also include support for the Desktop Management Task Force (DMTF) by supplying a DMI Agent, this however may be included after the availability of Chicago.

Chicago Tools

Chicago includes a number of built-in tools for network management, including NetWatcher (shown below). NetWatcher allows local and remote management of users connections to Chicago peer services. The tool shows all current connections to the Chicago system, who is connected and which files or printer is in use. It allows disconnection of the user, and also maintains an event log of key system events, log on, log off, system boot and shutdown, failed attempts to connects, and so on.



User	Computer	Shares	Open Files	Connecte
BOBT	\\BRENTE...	2	0	0D 0:6:2

IPC$
ADMIN$

Figure 58. NetWatcher Supports Local and Remote Monitoring of Chicago File and Printer Sharing

Printing Improvements

Chicago offers several changes in how printing is handled by Windows. The goals are to address requests made from our customers and from independent software and hardware vendors. We aimed to make improvements in three major areas:

- **Improved performance.**

  Chicago has a new 32-bit printing architecture which supports preemptive multitasking and improves overall performance.

- **Making it even easier to use.**

  Printing is easier now because of improvements made in Chicago's user interface. Chicago also features Plug and Play support for installing new printers.

- **Improved integration of network printing.**

  Network printing integration is better because of Chicago has extended the local printing architecture to the network environment. In addition, Chicago ties together installation enhancements to shared network printers.

This section of the Guide describes the printing architecture used in Chicago and discusses the different areas where printing has been improved over Windows 3.1.

Summary of Improvements over Windows 3.1

The primary improvements in printing for Chicago are:

- New 32-bit Print subsystem modeled after Windows NT providing smooth background printing
- Increased printing performance by decreasing time needed to return control to application through the use of enhanced metafile (EMF) spooling
- Support for over 800 different printer models (versus over 300 for Windows 3.1) through the development of new printer mini-drivers
- Support for PostScript Level II printers
- Spooling of MS-DOS–based application print jobs along with Windows–based applications and solves conflicts when MS-DOS and Windows–based applications try to print at the same time
- Image color matching support providing better WYSIWYG between color in images displayed on-screen and color generated on an output device

- Deferred printing for mobile computer users, allowing users to print while undocked and not connected to a printer, then automatically starting print job once docked into a docking station
- Simplified printer driver installation, configuration, ease of use, and ease of support, through new consolidated user interface
- System support for new bi-directional printers and ports providing improved I/O performance with new fast parallel ports (ECP) and error status reporting
- Better integration of network printing support including point-and-print support for automatic installation of printer drivers from Chicago, Windows NT, or Novell NetWare servers
- Plug and Play support for printers for simpler installation and configuration

32-bit Print Subsystem

Chicago features a 32-bit Print subsystem that includes a multi-threaded, preemptive spooler architecture providing improved printing performance, smoother background printing, and quicker "return to application" time once a print job is initiated by a user in an application. The architecture of the Print subsystem is compatible with the Windows NT 3.1 Print subsystem.

32-bit Preemptive Spooler

In Windows 3.1, print spooling functionality was handled by Print Manager and was supported by code in several different Windows components. In Chicago, the print spooler is implemented as a series of 32-bit virtual device drivers and consolidates the spooler functionality into a single architecture.

- **The new spooler provides smooth background printing.**
  In Windows 3.1, Print Manager passed a fixed amount of information down to the printer, whether the print was ready to receive it or not. If the printer wasn't ready to receive more data, the system would be suspended until the printer was ready. Unlike Print Manager, the Chicago spooler passes data to the printer only when the printer is ready to receive more information. This helps to reduce what often seemed like "jerkiness" when printing documents with Windows 3.1 Print Manager.
- **The new spooler provides quick "return to application" time.**
  Due to the smooth background printing, made possible by the new 32-bit print subsystem, Chicago spools enhanced metafiles (EMF) when printing from Windows–based applications rather than raw printer data to result in quicker return to application time. Once spooled, the EMF information is interpreted in the background by the printer driver, and output is then sent to the printer. For more details, see the following "Enhanced Metafile Spooling" section.

- **The new spooler is much more powerful and flexible.**
  It allows the user to select printer attributes on a per printer basis instead of requiring global printing attributes as in Windows 3.1. For example, each printer can have a different separator page and the option of printing direct printing via a queue.

Enhanced Metafile Spooling

EMF spooling results in quicker "return to application" time for returning control to the user after initiating a print job in a Windows–based application (Win16 or Win32).

Before discussing how EMFs fit into the printing architecture used by Chicago, it is worth reviewing how print jobs are handled by Windows 3.1. The improvements present in Chicago result in great printing performance over Windows 3.1.

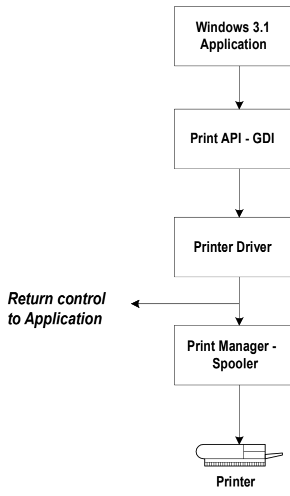


Figure 59. Spooler Relationship to Printing in Windows 3.1

In Windows 3.1, all interpretation of print API calls were handled by the Windows printer driver *before* the information was spooled to Print Manager. The interpretation of print information for printers was the most time-consuming operation in the print process. Postscript printers were not impacted by this as the printer driver sends high-level Page Description Language (PDL)-based information to the printer rather than raw image data, where it was interpreted by the printer itself. Users of non-Postscript printers saw a delay in "return to application" time under Windows 3.1 once the print job was initiated while the GDI print API calls are processed by the printer driver. Once the output image file was created by the printer driver, the Print Manager spooler took over and control was returned to the user's application. Background printing under Windows 3.1 often seemed choppy.

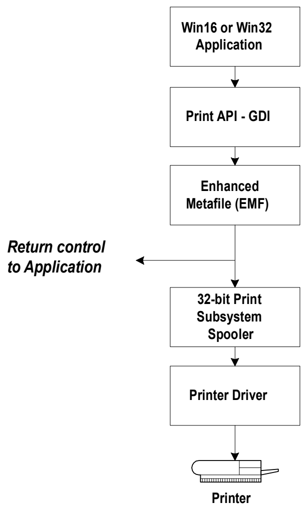

Win16 or Win32 Application
Print API - GDI
Enhanced Metafile (EMF)
Return control to Application
32-bit Print Subsystem Spooler
Printer Driver
Printer

Figure 60. Spooler Relationship to Printing in Chicago

Chicago greatly improves the "return to application" time by spooling high-level command information generated by the GDI print API, collectively referred to as an enhanced metafile, rather than spooling raw printer data generated by the printer driver. For example, if a document contains a solid black rectangle, the EMF would contain a command to draw a rectangle with the given dimensions, that should be filled in solid, with the color black. Once the EMF is created, control is returned to the user, and the EMF file is interpreted in the background by the 32-bit print subsystem spooler and sent to the printer driver. This results in control being returned to the user in significantly less time than having to wait for the print calls to be fully interpreted by the printer driver directly.

Improved Printing Support for MS-DOS–based Applications

Chicago improves on support for printing from an MS-DOS–based application in the Windows environment over that provided by Windows 3.1 by allowing MS-DOS-based applications to spool print jobs to the Chicago Print subsystem 32-bit spooler. With Windows 3.1, users printing from MS-DOS applications could not take advantage of the Windows–based spooling functionality offered by Print Manager, and encountered device contention issues when trying to print from MS-DOS–based applications at the same time as printing from Windows–based applications.

Chicago addresses the print issues in Windows 3.1 by incorporating the functionality for an MS-DOS–based application to spool directly to the 32-bit Chicago print spooler. This support is integrated into a print spooler virtual device, which takes the output destined for a printer port and first places it in the Chicago print spooler before sending the data to the printer. This functionality works with all existing MS-DOS–based applications, and results in quicker "return to application" through the use of the spooling mechanism. While MS-DOS–based applications do not benefit from EMF spooling, which is supported only for printing from Windows–based applications, users won't encounter device contention issues, but will benefit from smoother background printing and from improved printing performance in Chicago. The print spooling functionality for use with MS-DOS–based applications is automatically installed and configured and handling is transparent to the user.

Support for Deferred Printing

To benefit mobile computer users, the Chicago Print subsystem also features support for deferred printing. This capability allows users not connected to a printer to generate print jobs, which are stored on their local computers. This feature is handy for users working in a location away from the printer, and for users in the office who temporarily lose printer connections because of network or printer problems, for example. Mobile users can create print jobs from Windows–based applications (Win16 or Win32 applications) or MS-DOS–based applications while on the road, then print on a physical printer once reaching home or office. Items not immediately printed are held in the print queue until the user reconnects to a printer.

Image Color Matching Support

Chicago includes Image Color Matching (ICM) support, enabling applications to offer better consistency between the color of images displayed on the screen and the color of images generated by an output device.

Using technology licensed from Kodak, Chicago will include image color matching (ICM) support for display and printer drivers. ICM provides consistent (predictable) color rending from input, through monitor preview, to output. Applications that utilize ICM functionality enables portability of color information across applications that manipulate the graphic information, across users to provide consistent use of colors, and across platforms allowing color information to be easily moved to different systems where the ICM technology has been implemented.

Image color matching support in Chicago provides the following benefits to application vendors, which in turn result in benefits to users:

- Easily enable color-aware applications
- Allows for color *What You See Is What You Get* (WYSIWYG)
- Provides for consistent color output across devices, and

Since Windows 3.1 did not provide ICM support as part of the operating system or in an external driver, image color matching support was implemented in a proprietary manner by an application vendor—the burden was on the application vendor to properly map colors generated on a display device to the colors generated by a printer device. Chicago simplifies this process by including ICM support as part of the operating system, allowing application vendors to integrate ICM functionality into their applications, and thus take advantage of this new system service. To provide support for device-independent color matching, colors used in applications are tied to international (CIE-based) colorimetric standards, rather than in device-dependent form to specific hardware devices. The operating system will then do the appropriate color transformations to map the device-independent color representations to the colors supported by the physical device.

The key to ICM support is the use of a *profile*, which represents the color properties of a monitor, printer, or scanner device. The profile format used by the ICM support in Chicago, is the result of an industry consortium called *ColorSync 2.0* and is made up of many industry hardware vendors (including Kodak, Microsoft, Apple Computer Inc., Sun, and Silicon Graphics, among others) and industry standard-setting bodies. The ColorSync efforts provide for a consistent cross-platform color standardization process that will result in industry-wide standards for defining ICM properties of output and display devices.

Installing and Configuring a Printer

The first thing you will notice when looking at printer support in Chicago are some changes to the user interface. Print Manager and the Print icon in Control Panel are gone. Gone also is the confusion of which tool to use when you wanted to manage a print job, install a new printer, create a queue, or perform some other task related to printing. Chicago consolidates the printer and printing functions into a single location called the Printers Folder.

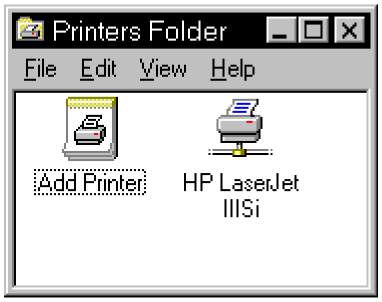


Figure 61. Chicago Printer Folders

The Printers Folder provides the user easy access to adding a new printer, configuring an existing printer, and managing print jobs.

Easy New Printer Setup

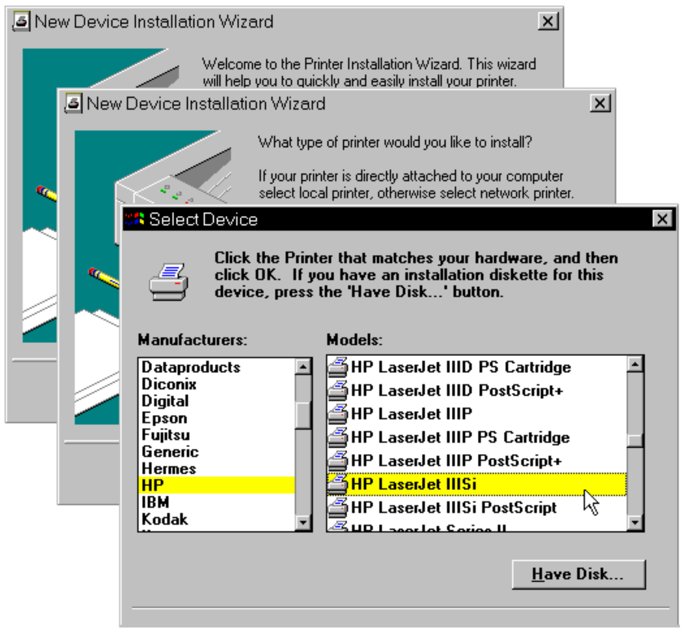
Chicago makes it easy to install new printers by supporting the following installation mechanisms:

- **Plug and Play Printer Detection**

  For Plug and Play printers, Chicago will automatically detect the printer at installation time, or during the Chicago boot process. The Plug and Play detection code will prompt the user for the appropriate driver files if they are not resident in the Windows directory.

- **New Device Installation Wizard**

  Chicago includes support for Wizards that walk the user through the printer installation process. Installing a printer is now even easier under Chicago, whether that printer is connected to the local PC, or shared on another PC on the network. The following figure shows the process that the New Device Installation Wizard uses for installing a printer.


New Device Installation Wizard
New Device Installation Wizard
Select Device
Manufacturers:
Models:
Have Disk...

Welcome to the Printer Installation Wizard. This wizard will help you to quickly and easily install your printer.

What type of printer would you like to install?

If your printer is directly attached to your computer select local printer, otherwise select network printer.

Click the Printer that matches your hardware, and then click OK. If you have an installation diskette for this device, press the 'Have Disk...' button.

Manufacturers:	Models:
Dataproducts	HP LaserJet IIID PS Cartridge
Diconix	HP LaserJet IIID PostScript+
Digital	HP LaserJet IIIP
Epson	HP LaserJet IIIP PS Cartridge
Fujitsu	HP LaserJet IIIP PostScript+
Generic	HP LaserJet IIISi
Hermes	HP LaserJet IIISi PostScript
HP	
IBM	
Kodak	

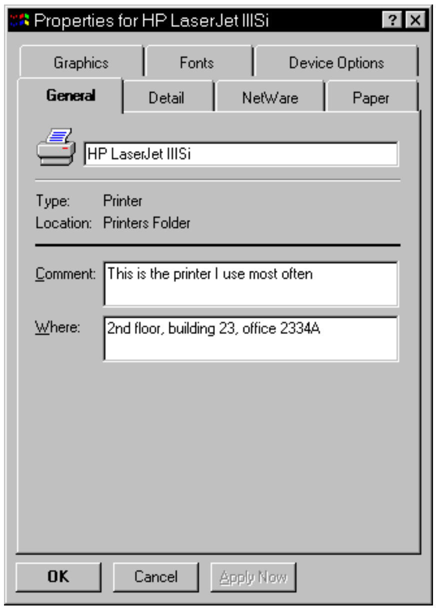
Figure 62. New Device Installation Wizard Walks User Through Installing a Printer

- **Connecting to a Network Printer—"Point and Print" Printing**

  Chicago makes it easy for users to connect to and use a printer shared on another Chicago PC, a Novell NetWare server, or a Windows NT Server. If a user connects to a printer shared on another Chicago PC, Chicago will automatically copy down and install the proper driver for the shared printer from the remote Chicago PC. Chicago also supports installing the appropriate printer drivers on a Novell NetWare server or Windows NT Server for automatic downloading and installation of the printer driver. This allows users to simply connect to the remote printer and once the drivers have been copied from the network and installed on the local Chicago PC, begin printing. Network Point and Print printing is discussed in the Chicago Networking section of this guide.

Configuring a Printer

Configuring a printer in Chicago is also greatly simplified over Windows 3.1. All printer configuration is consolidated into a single property sheet for the printer, and can be accessed from the Printers folder. The property sheet provides common access to printer parameters such as the printer port (or network path) the printer is connected to, the paper options for the printer, the fonts built into the printer, and device options specific to the printer model.



HP LaserJet IIISi

Type: Printer
Location: Printers Folder

Comment: This is the printer I use most often

Where: 2nd floor, building 23, office 2334A

Figure 63. Sample Properties for the Hewlett-Packard LaserJet IIIsi Printer

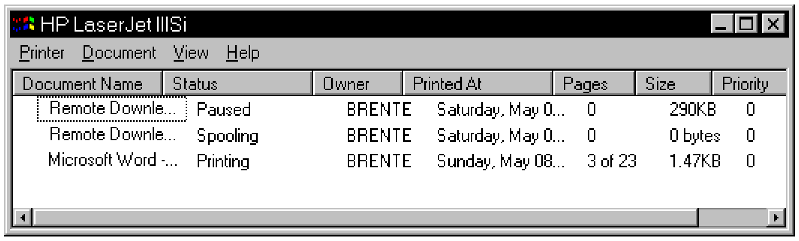
To further simplify printer configuration, Chicago will support bi-directional communications between compatible printers and printer ports, allowing Chicago to query the characteristics and configuration options directly from the printer. This allows the printer driver to be automatically configured to exactly match the configuration of the printer including the amount of memory installed, the paper options available, and fonts installed in the printer.

Managing Print Jobs

Managing print jobs is improved in Chicago over the capabilities offered by Print Manager in Windows 3.1 and Windows for Workgroups. Improvements provided in Chicago include:

- **Direct Integration with the Chicago User Interface**

  The Printers folder serves as the centralized location for interacting or configuring with printer devices. Switching to Details view will show additional information about the number of jobs presently residing in the printer. Opening the given printer will show detailed information about the contents of active print jobs or jobs that are waiting in the queue including the name of the document, the status of the document, the owner of the document, when the document was submitted to the print queue, the number of pages in the document (when printing, the status of the print job down to the page that is being printed, is displayed), the size of the document, and the priority of the print job.



Document Name	Status	Owner	Printed At	Pages	Size	Priority
Remote Downle...	Paused	BRENTE	Saturday, May 0...	0	290KB	0
Remote Downle...	Spooling	BRENTE	Saturday, May 0...	0	0 bytes	0
Microsoft Word -...	Printing	BRENTE	Sunday, May 08...	3 of 23	1.47KB	0

Figure 64. Detailed Remote Print Queue Status

- **Ability to Manage Print Jobs Locally and Remotely**

  Under Chicago, a user has the ability to pause or cancel printing of print jobs residing in a remote Chicago print queue. Under Windows for Workgroups, for example, a user needed to physically walk over to the remote PC to cancel any printing operations. In addition, if you are given administrator access to the remote Chicago PC that is sharing a printer, you have the ability to remotely manage and administer the print queue with the same UI and functionality available for a locally connected printer. It is no longer necessary to walk over to the remote machine where the queue resides, in order to terminate print jobs or to resume the printer if an error occurs.

Network Printing Improvements

Chicago provides improved support over Windows 3.1 for printing in a networked environment. These enhancements include:

- **Network point-and-print functionality**

  Under Chicago, users can print to a shared network printer connected to a computer running Chicago, Windows NT Advanced Server, or Novell NetWare, and have the appropriate printer driver automatically copied down from the remote computer and configured on the local Chicago computer. This simplifies the printer installation process, and ensures that the proper printer driver is installed to match the remote printer.

- **Microsoft Print Server for NetWare**

  Chicago provides a 32-bit Netware-compatible print server, offering functionality equivalent to the Novell Pserver utility and allows a computer running Chicago to despool print jobs from the queue on a NetWare server. As a 32-bit application, the Microsoft Print Server for NetWare provides smooth background processing and printing of jobs, as well as robust operation.

- **Remote Administration of Print Jobs**

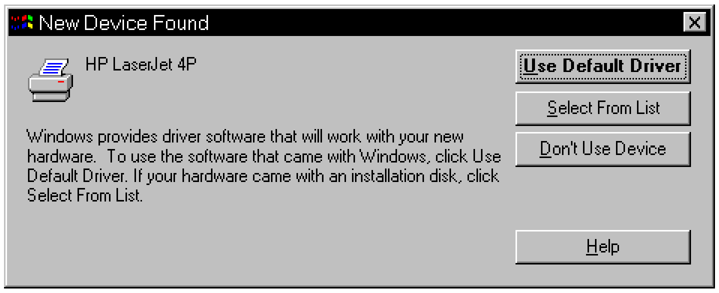
  Chicago provides full remote administration of print jobs for shared printers on another computer running Chicago. With the appropriate access privileges, these operations include the ability to hold a print job, cancel a print job, or resume printing when the print queue is paused.

More information about network printing enhancements present in Chicago is provided in the Networking section of this guide.

Plug and Play Support

Installing and configuring printers in Chicago is greatly simplified over Windows 3.1. As with other components of the Chicago system, setting up new printers in Chicago benefits from Plug and Play capabilities. Chicago detects Plug and Play compatible printers that return device ID values as described in the IEEE 1284 Specification through bi-directional parallel communications. Bi-directional parallel communications with the printer will also aid in the query of other physical attributes of the device.

Chicago detects a Plug and Play printer in one of several ways—when Chicago is first installed on a user's PC, during the boot cycle each time Chicago is started, or when a user explicitly requests a detection to be made.



Windows provides driver software that will work with your new hardware. To use the software that came with Windows, click Use Default Driver. If your hardware came with an installation disk, click Select From List.

Figure 65. New Device Found Dialog Box Showing Detection of Plug and Play Printer

Plug and Play for printers works this way:

When Chicago is first installed on a user's PC and when Chicago starts up, the Plug and Play detection code attempts to identify a printer connected to a bi-directional communications port. If the printer connected is not presently configured in the Chicago system, the user is asked as to whether the printer should be installed. If the user says yes and the appropriate printer driver is already present on the system, Chicago automatically installs and configures the driver for the new printer. If the printer driver is not already present, Chicago prompts the user for the appropriate Chicago Setup and Installation disk. If the printer is not recognized by the Chicago system, the user is prompted to insert a disk containing the printer driver provided by the printer manufacturer.

Examples of Plug and Play-Compatible Printers

While there will be much broader support and demand for Plug and Play printers at the time Chicago ships, several printers on the market today provide varying degrees of Plug and Play awareness. For example, the Hewlett-Packard's LaserJet 4P, 4MP, and 4ML, which are presently shipping, are Plug and Play peripherals.

Try It!

To see how the improvements made to printing support in Chicago will help users that print from MS-DOS and Windows-based applications, you've got to try it!

Mouse

Quick Return to Application Time

To demonstrate the improved quick return to application time for printing from MS-DOS or Windows-based applications, try the following:

- Under Windows 3.1, start Print Manager. Turn off background printing if supported by your application (for example, Word for Windows version 6.0). Print from your application - how long did it take before control was returned? Perform the same task under Chicago - how long did it take before control was returned?
- Quick return to application time is also evident when printing from MS-DOS–based applications. Try printing from an MS-DOS–based application under both Windows 3.1 and Chicago - how long did it take before control was returned?

Spooling from an MS-DOS–based Application

In addition to providing quick return to application time when printing from MS-DOS–based applications, print jobs generated by MS-DOS–based applications show up in the Chicago print queue and can be manipulated like print jobs from Windows–based applications. To see this, try the following.

- Pause the Chicago print queue for your printer and then print from an MS-DOS–based application and note that it shows up in the print queue.

Plug and Play Support

To try Plug and Play support for printers with Chicago, connect one of the supported Hewlett-Packard LaserJet printers (4P, 4MP, and 4ML) that is Plug and Play aware to your computer before starting Chicago. During the Chicago boot process, the Plug and Play printer will automatically be detected and will prompt you to install the appropriate printer driver.

Communications

Chicago features a new 32-bit communications subsystem that provides higher throughput, better reliability, and greater device independence for communications operations than Windows 3.1. The new communications subsystem serves as the underlying architecture on which Chicago provides communication services for supporting remote network access, Microsoft At Work fax services, access to on-line information services, and remote access to mail.

The communications architecture addresses problems that users have encountered with communications support in Windows 3.1 to provide a powerful, robust, and flexible communications architecture.

Summary of Improvements over Windows 3.1

Changes made in the kernel and communications architecture in Chicago provide improvements and benefits to the Windows 3.1 user including:

- Robust and reliable high baud rate communications throughput
- Better multitasking of communications applications
- Simpler centralized setup and configuration
- Broader device support, and
- Better support for sharing communication devices on a PC (e.g., modems) among different communication applications

Communications Architecture

Around the time when Windows 3.0 was first developed, 2400 baud modems were the mainstream and 9600 baud modems were just becoming affordable. Windows was able to handle receiving data at these relatively slow rates without much difficulty. However, as mechanisms to transfer communications information at faster rates (e.g., higher baud rates or the through the use of data compression) are becoming more popular, the communications architecture of Windows needed to be examined closely.

Around the time when Windows 3.1 shipped, 9600 baud modems were extremely popular, and communications under Windows 3.1 had many barriers that limited the overall effectiveness of reliable high data throughput and support for multitasking when running communications applications. These barriers included high interrupt latency and overhead affecting high speed communications, and a monolithic driver architecture that made it necessary for some third-parties to replace the communications driver provided with Windows to allow their devices to run efficiently in the system.

Chicago greatly improves upon the Windows 3.1 architecture to support communications applications and supports high speed communications, as well as a modular communications architecture to allow third-parties and communications device manufacturers to easily plug in new communications device drivers. This section describes the communications architecture used in Chicago.

Chicago Communication Goals

The goals of communications support in Chicago are focused around supporting an architecture that delivers better performance than Windows 3.1, and supports ease-of-use enhancements through Plug and Play communications. The communications architecture of Chicago delivers the following performance benefits over Windows 3.1:

- **High-speed reliability**

  Chicago supports reliable high-speed communications by keeping up with data coming in from the communications port, and thus resulting in no lost characters due to interrupt latency. In addition, the use of a 32-bit protected mode file system and network architecture results in less impact on the communications system by reducing required mode transitions and interrupt latency.

- **Higher data throughput**

  The 32-bit communications subsystem leverages the preemptive multitasking architecture of Chicago to provide better responsiveness to communications applications, and thus supporting higher data throughput. Communications transfers in 32-bit applications are not as affected by other tasks running in the system as Win16–based applications under Windows 3.1.

- **Provide support for time critical protocols**

  Chicago's communications architecture provides support for time critical protocols and allows for real-time serial device control.

The Plug and Play initiative provides ease-of-use enhancements system-wide in Chicago and communications support is no different. Plug and Play support for communications delivers:

- **Broad device support**

  Chicago features a new communication driver architecture that makes it easier for third-parties to extend the communications support provided as part of the operating system, without sacrificing functionality or stability. In addition, the new communications architecture features APIs that are extended to support more robust communications devices beyond base RS-232 devices (e.g., ISDN).

- **Easy to install and use communication devices**

    Chicago features centralized modem installation and configuration support to simplify setup for end-users, and simplify communications development efforts for application developers. Chicago leverages the use of a single universal modem driver (UNIMODEM) to provide a consistent mechanism for communicating with modem devices. Chicago also provides detection support for Plug and Play modems. It also provides support for existing hardware by including mechanisms for detecting legacy modems.

- **Enables device sharing among communications applications**

    Through the use of the Telephony API (TAPI), Chicago provides consistent device-independent access to control communication devices for operations such as dialing and answering incoming calls. Arbitration for sharing of communication ports and devices is also handled through TAPI. For example, while Chicago remote access services is waiting for an incoming call, a TAPI-aware fax communications application can send an outgoing fax without having to first terminate the already running communications application.

To further understand the improvements related to the new 32-bit communications subsystem in Chicago, we'll examine the components that make up the communications support.

Chicago Kernel Improvements

When data is coming into the system from a serial communications port, an interrupt occurs telling the system that a piece of data has just been received. Under Windows 3.1, if information is being received at a high rate it was possible that the system could not keep up with the incoming data, thus resulting in errors or lost information at the port.

What is unique about serial communications I/O is that one interrupt occurs on the system for *each* incoming character, versus disk or network I/O in which they manipulate blocks of information at a time. The burden on the communications driver to keep up is quite high.

To support high speed throughput of information from a communications device, the system must be able to respond quickly to incoming data. In Windows 3.1, real-mode drivers sometimes disabled system-wide interrupts for a "long" period of time (usually in terms of milliseconds), during which no incoming information can be received.

To directly address the issue of supporting higher sustained communications throughput, the Chicago development team focused on areas in the Windows kernel that resulted in periods of time that interrupts were disabled by the system. In Windows 3.1, the Windows kernel and other components was limited to reliable serial communications at rates of 9,600 bps or slightly higher (dependent upon CPU speed), due to high interrupt latency or other systems design limitations. In addition, the use of real-mode file system and networking drivers would block the system when Windows 3.1 had to execute real-mode code, thus preventing the system from being able to keep up with incoming data.

To improve performance and the rate at which the system can accept incoming data reliably, the Chicago team reduced code that can only be used by one process at a time (critical sections), and reduced interrupt latency in the core system. In addition, the use of new 32-bit protect mode components for the implementation of the file system and network subsystem also helped to improve the system responsiveness. Chicago is now truly limited in baud rate only by the hardware characteristics of the PC such as the processor speed, and type of communications port.

Driver Architecture

The communications subsystem consists of a modular 32-bit protect mode architecture with *new communications drivers*. VCOMM is a new layer that provides protected mode services that allow Windows-based applications and device drivers to use ports and modems. To conserve system resources, communications device drivers are loaded into memory only when in use by applications. Also, VCOMM uses the new Chicago Plug and Play services to assist with configuration and installation of communications devices.

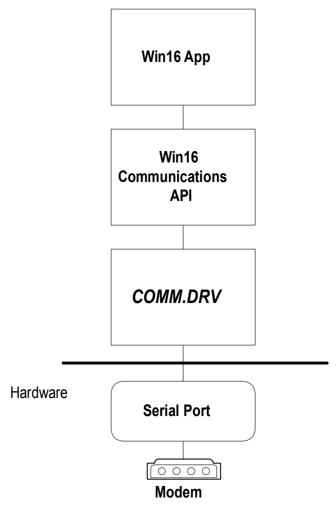


Figure 66. Windows 3.1 Communications Architecture

Windows 3.1 uses a monolithic communications driver, COMM.DRV, that provides an API interface for Windows–based applications to interact with communications devices and the code that serves as the communications port driver. The monolithic approach made it necessary to completely replace the Windows communication driver if new functionality was required by a hardware device. Figure 66 shows the relationship between the COMM.DRV driver and the hardware device in Windows 3.1.

Chicago provides a more flexible communications architecture than Windows 3.1, separating communications operations into three primary areas—Win32 communication APIs and TAPI, the universal modem driver, and communication port drivers.

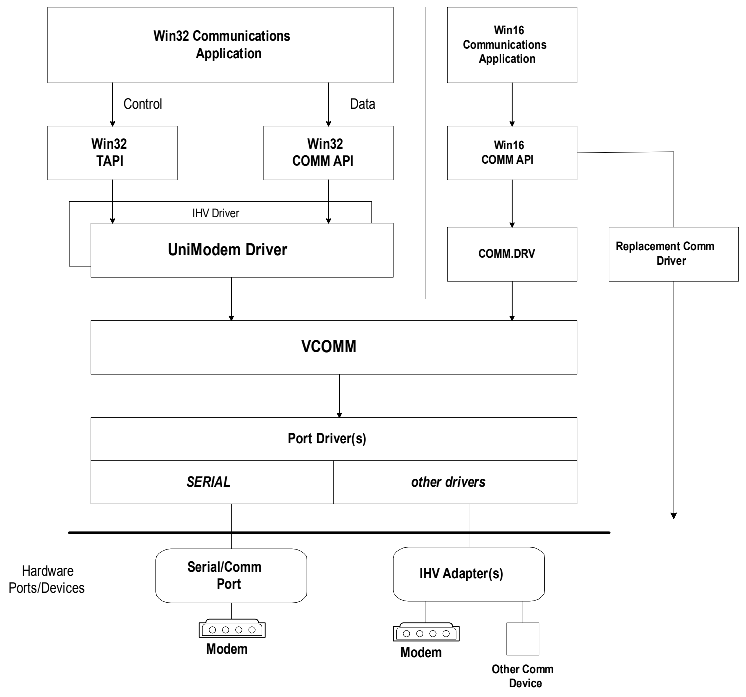


Figure 67. Chicago Communications Architecture

Figure 67 shows the relationship between the VCOMM communications driver and the port drivers to communicate with hardware devices. The flow path for a Win16–based application is also illustrated to show how compatibility is maintained for existing Windows–based applications. Compatibility is maintained for IHVs and ISVs that replace the Windows 3.1 COMM.DRV driver, however the vendor-specific communications driver communicates directly with the I/O port, rather than going through VCOMM.

A description of the primary areas that make up the architecture are described below.

- **Win32 Communication APIs and TAPI**

  Chicago's Win32 Communications APIs provide an interface to use modems and communications devices in a device-independent fashion. Applications will call the Win32 Communication APIs to configure modems and perform data I/O through them. Through the Telephony API, applications will be able to control modems for operations such as dialing, answering, or hanging up a connection, in a standard way. TAPI-aware communication applications no longer need to provide their own modem support list, as interaction with a modem is now centralized by Chicago. The communications functionality provided with Chicago utilizes these services.

- **Universal Modem Driver**

  Also new in Chicago is the universal modem driver, UniModem, which is a layer for providing services for data and fax modems, and voice. Users and application developers will not have to learn or maintain difficult modem “AT” commands to dial, answer, and configure modems. Rather, UniModem does these tasks automatically, using mini-drivers written by modem hardware vendors. Application developers can utilize TAPI to perform modem control operations in a modem-independent manner.

- **Port Drivers**

  Port drivers are specifically responsible for communicating with I/O ports. I/O ports are accessed through the VCOMM driver and provide a layered approach to device communications. For example, Chicago will provide a port driver to communicate with serial communications and parallel ports, and third-parties and IHVs can provide port drivers to communicate with their own hardware adapter such as multiport communications adapters. With the port driver model in Chicago, it will no longer be necessary for third-parties to replace the communications subsystem as they did in Windows 3.1.

Telephony API (TAPI)

The Windows Telephony API is part of the Microsoft Windows Open Services Architecture (WOSA), which provides a single set of open-ended interfaces to enterprise computing services. WOSA encompasses a number of APIs, providing applications and corporate developers with an open set of interfaces to which applications can be written and accessed. WOSA also includes services for data access, messaging, software licensing, connectivity and financial services.

WOSA services such as the Windows Telephony API consist of two interfaces. Developers write to an applications programming interface (API). The other interface, referred to as the service provider interface (SPI), is used to establish the connection to the specific telephone network. This model is similar to the computer industry model whereby printer manufacturers provide printer drivers for Windows-based applications. Figure 68 shows the relationship between the “front-end” Windows Telephony API and the “back-end” Windows Telephony SPI.

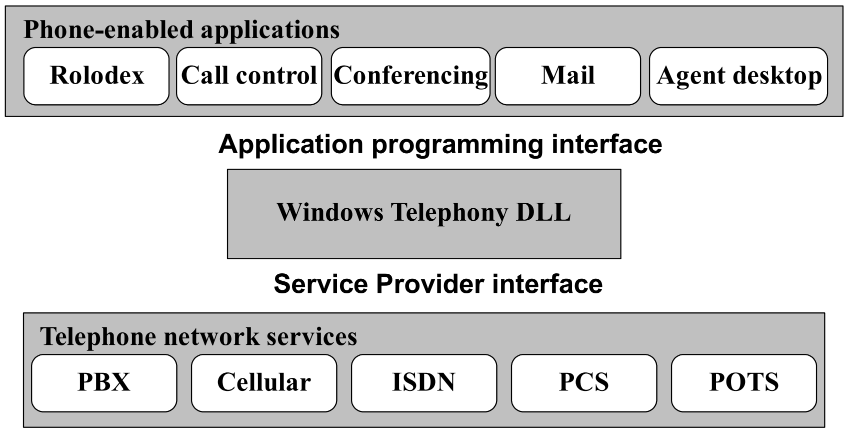

Phone-enabled applications
Rolodex
Call control
Conferencing
Mail
Agent desktop
Application programming interface
Windows Telephony DLL
Service Provider interface
Telephone network services
PBX
Cellular
ISDN
PCS
POTS

Figure 68. Windows Seamlessly Integrates Applications and Telephone Networks

The Windows Telephony API provides a standard way for communication applications to control telephony functions for data, fax, and voice calls. This includes such basic functions as establishing, answering and terminating a call. It also includes supplementary functions, such as hold, transfer, conference and call park found in PBXs and other phone systems. The API also provides access to features that are specific to certain service providers, with built-in extensibility to accommodate future telephony features and networks as they become available.

Through the use of the TAPI services, applications that support communication services have a device-independent means for interacting with communications devices. TAPI also provides a common access mechanism for requesting the use of communication ports and communication devices, thus providing a means for communications-based applications to share a single modem (voice, fax, or data) in the computer.

Chicago includes TAPI support in the base operating system, thus allowing application developers to leverage this functionality in their Chicago-aware applications. In addition, all communication components included as part of Chicago are TAPI clients.

Better Sharing of Communication Devices Between Communication Applications and Services

Through the TAPI interface, communications applications can ask for access to the modem or telephone device, allowing the Chicago Communications subsystem to arbitrate device contention and allow applications to share a communications device in a cooperative manner.

Win32–based applications can utilize TAPI functionality to allow applications to be able to make outgoing calls, while others are waiting for inbound calls. For example, while a remote network access service is configured for auto-answer mode and is waiting for an incoming call, a Win32–based communications application can call the TAPI services to request the use of the modem and perform an outgoing call. Of course, only one call can be performed at a time, but users no longer have to terminate other applications that are using a communications port in order to run a different communication. The TAPI services arbitrate requests to share communication ports and devices.

Centralized Modem Setup and Configuration

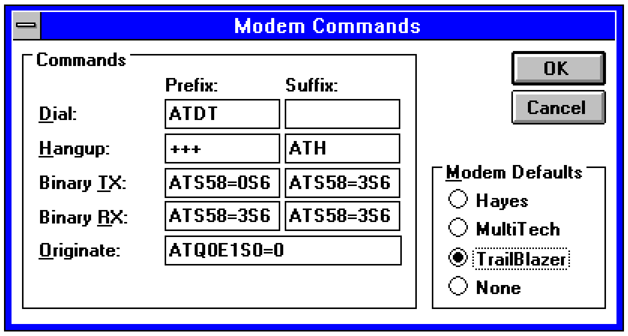
Support for installing and configuring a modem under Chicago is greatly simplified over Windows 3.1. No longer is it necessary to configure each individual communications program for the proper serial port, modem type, and other related modem configuration parameters. Chicago provides central configuration of communications devices through an icon in Control Panel. Win32–based applications that take advantage of the TAPI services implemented in Chicago can completely leverage the user's configuration of their communications hardware, making subsequent configuration of communications-based applications easy.

Chicago brings the following benefits to modem configuration:

- Easy modem configuration for use by entire system for new communications applications
- Centralized communications port status and configuration
- Supported by TAPI & Win32 Communication APIs
- Support for 100+ modems in Chicago

Modem Configuration in Windows 3.1

To understand the implications of improved modem support in Chicago, let's first examine the issue of modem configuration under Windows 3.1. Under Windows 3.1, when a user adds a new communications program to their computer, the user must first configure the application to communicate with their modem by specifying the COM port to use, the type of modem they have, in addition to other communication parameters. Communications and modem configuration is either handled by the applications vendor and specified as a series of default modem "AT" command statements, or it is up to the user to read through his/her modem manual and type in the appropriate command strings. Figure 69 shows the Modem Commands dialog box in Terminal provided with Windows 3.1. Furthermore, given the number of modems available on the market, many Windows 3.1–based communications applications support a limited set of recognized modems because of the increased burden on the applications developer to provide this support.



Commands

	Prefix:	Suffix:
Dial:	ATDT	
Hangup:	+++	ATH
Binary TX:	ATS58=0S6	ATS58=3S6
Binary RX:	ATS58=3S6	ATS58=3S6
Originate:	ATQ0E1S0=0	

Modem Defaults: Hayes / MultiTech / [✓ TrailBlazer] / None

Figure 69. Modem Configuration in Windows 3.1 Terminal

Modem Configuration in Chicago

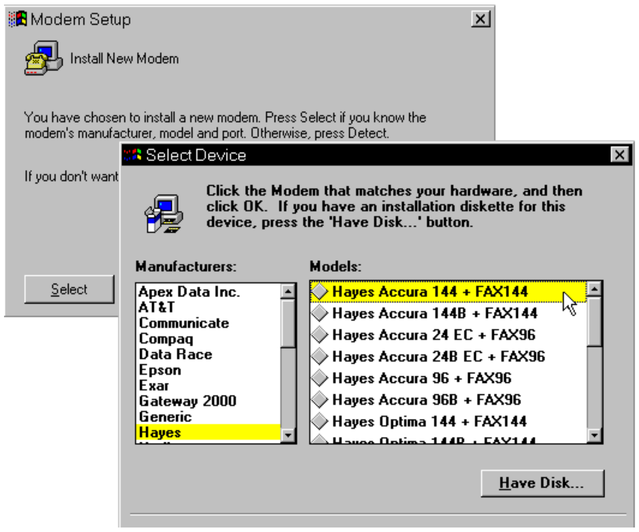
As with support for printers, the use of modems in Chicago is centralized by the operating system. When a user first installs Chicago, the user is prompted to detect or identify the modem device that they have connected to or installed in their computer. Once a modem has been selected and configured, any communications application that supports TAPI services can interact with the modem in a device-independent way. Users no longer need to know or understand complicated "AT" command sequences to customize their communications application.

Configuring a modem under Chicago is as easy as performing three simple steps: identifying the new modem device, configuring the modem device, and configuring the Telephony services.

Identifying the New Modem Device

Modems

If the user doesn't select a Modem when Chicago is first installed, a new modem can be identified from the Modems control panel icon. When the Modems dialog box is displayed, the user can either select a modem from the list of known manufacturers and models, or can choose the option to have Chicago detect the modem connected to the PC. The detect option makes it easy for users by using Plug and Play detection or querying the modem to configure the correct device. If Chicago can not detect the device, then the user can manually select the proper modem to use.


Modem Setup

Install New Modem

You have chosen to install a new modem. Press Select if you know the modem's manufacturer, model and port. Otherwise, press Detect.

If you don't want

Select

Select Device

Click the Modem that matches your hardware, and then click OK. If you have an installation diskette for this device, press the 'Have Disk...' button.

Manufacturers:

- Apex Data Inc.
- AT&T
- Communicate
- Compaq
- Data Race
- Epson
- Exar
- Gateway 2000
- Generic
- Hayes

Models:

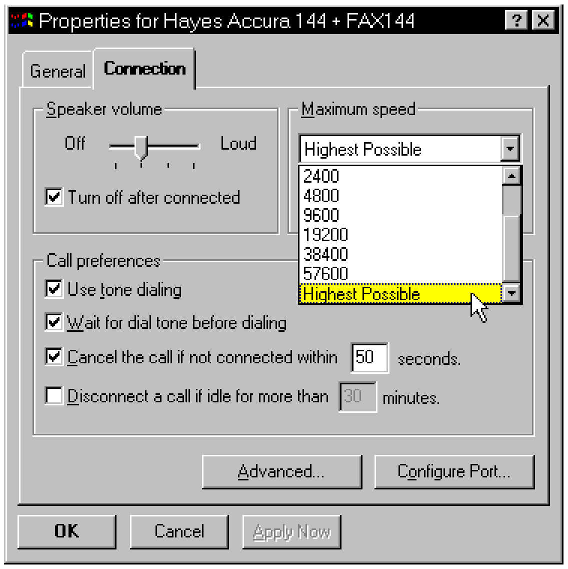
- Hayes Accura 144 + FAX144
- Hayes Accura 144B + FAX144
- Hayes Accura 24 EC + FAX96
- Hayes Accura 24B EC + FAX96
- Hayes Accura 96 + FAX96
- Hayes Accura 96B + FAX96
- Hayes Optima 144 + FAX144

Have Disk...

Figure 70. Selecting a Modem Device

Configuring a Modem Device

Once the proper modem has been selected, the user can optionally change the properties for the device to set configuration parameters such as the volume for the modem speaker, the time to wait for the remote computer to answer the call, and the maximum baud rate to use. (The maximum baud rate is limited by the speed of the PC's CPU and the speed supported by the communications port.)


Properties for Hayes Accura 144 + FAX144

General | Connection

Speaker volume: Off — Loud

[✓] Turn off after connected

Maximum speed: Highest Possible

- 2400
- 4800
- 9600
- 19200
- 38400
- 57600
- Highest Possible

Call preferences

[✓] Use tone dialing

[✓] Wait for dial tone before dialing

[✓] Cancel the call if not connected within 50 seconds.

[ ] Disconnect a call if idle for more than 30 minutes.

Advanced... | Configure Port...

OK | Cancel | Apply Now

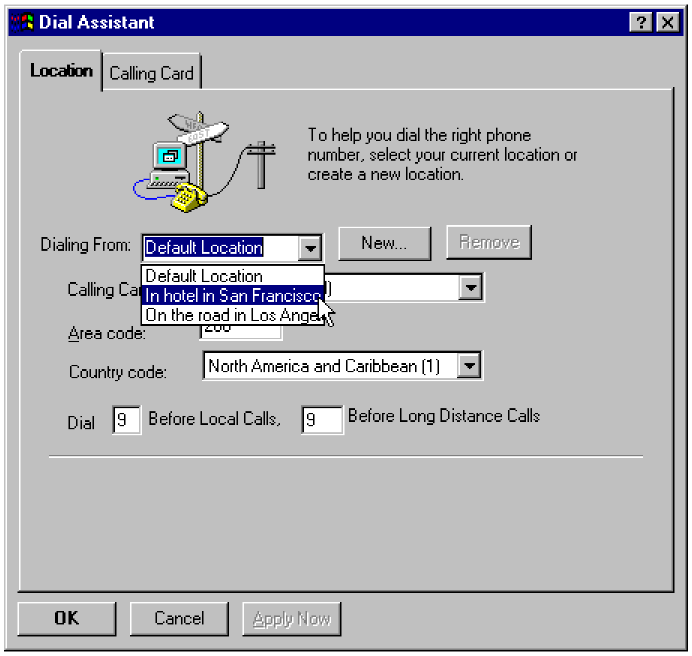
Figure 71. Sample Modem Property Sheet

Configuring Telephony Services

In addition to configuring the modem device, the user configures telephony services to identify the various dialing parameters associated with the different locations where the computer will be used.

For each location where the PC will be used, information is stored for use by TAPI-aware applications to simplify dialing a local call, a long distance call, the area code for the location for use in determining whether the call is inside or outside the calling area code, and calling card information. For a desktop PC, the location would commonly use the "default location" or perhaps change the default name to "in the office," whereas a mobile user would add several different locations to match where the laptop computer is commonly used.

For example, a mobile user may use the computer in the office, on the road, or in a remote city as part of his/her business needs. The following figure shows several different locations selected.


Dial Assistant

Location | Calling Card

To help you dial the right phone number, select your current location or create a new location.

Dialing From: Default Location

Default Location
In hotel in San Francisco
On the road in Los Ange

Calling Ca

Area code: 206

Country code: North America and Caribbean (1)

Dial 9 Before Local Calls, 9 Before Long Distance Calls

Figure 72. Dial Assistant Property Sheet for Configuring Location Information

Improved Device/Hardware Support

Chicago provides improved communications device and hardware support over Windows 3.1. A few areas of improvement are discussed below.

16550A UART FIFO Support

Chicago will provide greater robustness and performance at high baud rates for MS-DOS-based and Windows-based communications applications using local serial ports with 16550A compatible UARTs. The 16550A UART contains a 16 byte FIFO buffer to prevent character overflow due to interrupt latency, and helps to reduce interrupt overhead overall. The Windows 3.1 communications driver did not fully support the use of the 16550A UART, requiring some third-party communications vendors to replace the Windows 3.1-provided communications driver. Communications in Chicago should alleviate the need for third-party vendors to replace communications driver components.

More Ports Supported

Windows 3.1 imposed a limit to the number of logical names that could be used to address or refer to serial communication ports and to parallel ports, of 9 serial ports and 4 parallel ports. The communication APIs present in Chicago have been enhanced to support the same number of logical ports as MS-DOS, which is 128 serial ports, and 128 parallel ports. This limit inhibited the use of multi-port serial devices in Windows 3.1. The actual limitations to the number of ports usable is still based on the physical number of ports available to the system.

Support for Future Parallel Modems

Chicago also provides support for Enhanced Capabilities Ports (ECP) to facilitate higher speed communications than is possible over a serial device. This support allows the use of future parallel port modems.

Plug and Play Support

Plug and Play support for communications devices in Chicago facilitates the detection of connected modem devices and assigning of system resources (e.g., IRQs and I/O addresses for communication ports), simplifying configuration and setup. In addition to Plug and Play detection, Chicago provides for manual detection of non-Plug and Play communication devices such as modems. Since there is presently no standard for obtaining device information using the “AT” modem command strings, detection of legacy modems is handled by performing a manual query of the modem device and checking information returned against a database of known modem information. Microsoft is working with other leading industry manufacturers to standardize the modem command set as part of a Telecommunications Industry Association (TIA) proposed standard called IS-131. When this proposal is adopted, Chicago will support the standardized command set and this will aid detection of legacy modems.

Modems

Plug and Play detection for external modems requires new firmware in the modem to return the required Plug and Play ID information, while internal Plug and Play modems utilize the ISA Plug and Play specification. Chicago supports the use of PCMCIA communication devices as part of the Plug and Play services for the PCMCIA specification. Some modem manufacturers will improve their communications product offerings by revising their existing modem lines, whereas others will produce a new line of Plug and Play modems. Detection for Plug and Play serial devices such as modems is handled when Chicago is initially installed, during the Chicago boot process, or when a new modem device is connected to the system. As with other Plug and Play devices, the user is notified that the new device has been detected and is asked to confirm the installation and configuration of the device.

Support for legacy modems is provided by using device-specific information about a modem to provide a manual detection mechanism, or providing a list of supported modems from which a user can choose the appropriate one. Once the modem is identified for the system, it is available for use by TAPI-enabled communication applications including remote access services, Microsoft At Work fax services, and the new HyperTerminal communications application.

New Communications Application: HyperTerminal

Chicago includes a new 32-bit communications application called HyperTerminal that demonstrates what it's like to be a good communication application under Chicago. HyperTerminal offers the same base communication capabilities as Terminal included with Windows 3.1, but integrates well with the Chicago UI and demonstrates how the Win32 communication APIs and TAPI services support more flexible communications applications than Windows–based applications under Windows 3.1.

Good communication applications under Chicago will utilize the following services and capabilities to offer a more robust and powerful solution:

- Win32–based application that uses the Win32 Communication APIs.
- Internal architecture that uses multiple threads of execution to provide good responsiveness to the user and great error-free high speed communications. Multiple threads allows for full preemptive multitasking of communication tasks, supporting concurrent interaction with the user, downloading of remote data, and display of communication status.
- Support for TAPI services for making remote connections and controlling the modem device.

Try It!

To see how the improvements made to communications support in Chicago will help users of existing MS-DOS and Windows-based communications applications, you’ve got to try it!

Background Multitasking of Communications Applications

To see how support for communications application in Chicago is improved over Windows 3.1, try the following under both Windows 3.1 and under Chicago:

- Run an MS-DOS–based communications application in the background with other foreground activities.
- Run a Win16–based communications application and perform other CPU or disk intensive tasks, such as copying files, accessing a network, or accessing/formatting a floppy disk.
- Run the 32-bit HyperTerminal communications application and perform other CPU or disk intensive tasks, such as copying files, accessing a network, or accessing/formatting a floppy disk.

Mobile Computing Services

As personal computing moves beyond its traditional fetters of a single desktop, Microsoft is committed to leading the market as the critical catalyst in this change. With Windows "Chicago," Microsoft is in a unique position to deliver system services and end-user functionality that takes mobile computing to the next level of opportunity.

Vision of Chicago Mobile Computing

The goal for mobility in Chicago is to allow users—wherever they are and whatever computing they want to do—to do it easily. Our strategy for delivering on this vision is based on the following tenets:

- **Mobile computing encompasses anyone that moves away from their desktop PC and wants computing capabilities.** It includes everyone, from the people who moves from meeting to meeting in an office building, to those who shuttle between their homes and offices, to those who have no office at all and move from customer site to customer site.
- **The tasks people want to do away from their desks are fundamentally similar to those that they want to do on their desktop.** On desktop PCs, people want to draft a memo, review a budget spreadsheet, browse email, peruse a presentation on the network, look at their schedule for the day. Away from their office and desktop PC, people want to continue browsing email, drafting a memo, or perusing a presentation up on the network.
- **The "mobile" computer environment is fundamentally different than the desktop environment.** When users move away from their desktop PC, their computing environment changes dramatically. Portable PC users are operating in a power constrained environment with a video display often half the size of their desktop display. Their hardware changes regularly as they plug in/unplug different components to deal with the task at hand. They can't participate in their workgroup so they can't access a file on a server or receive email. In this way, the mobile computing environment can be constrictive to users.

Chicago Mobile Framework

Mobile computing restrictions stem from two problems:

- **Communicating**

  At their desk, users have a wide array of communication capabilities to keep them connected to the people in their network workgroup and people outside of their workgroup. They have access to the LAN and all its services such as email, file sharing, and print sharing. There is a phone, a fax machine, and perhaps a shared modem close at hand. When they leave their desks, these users become communication islands. They are cut off from their workgroup and all its LAN services. Phones, faxes, and modems are not readily available. Being mobile is a constant struggle to stay in touch.

To realize seamless mobility, users must be able to easily communicate regardless of location whether it be in the office, at a customer site, or at home. Communication is between their portable and their desktop PC, between themselves and the rest of their workgroup, and between themselves and the broader community of PC users.

Chicago provides compelling end-user communication capabilities and an open, extensible set of services for applications to establish these connections.

- **Adapting to hardware changes and differences**

  Desktop PCs operate in a fairly constant environment. Mobile computers do not. For example, docking station owners usually change video resolution, pointing device, and network state every time they change location (for example, when they undock). To change locations means tweaking configuration files, contending with error messages and restarting their computer a lot. Being mobile is a constant struggle to get the hardware to adapt to the new conditions the user is computing in.

  A portable PC user's hardware configuration is constantly changing as he or she move from location to location. To achieve seamless mobility, these changes in hardware must be seamless to the user.

  Microsoft's Plug and Play architecture, defined in partnership with other industry leaders like Intel and Compaq, provides an infrastructure to effectively tackle these problems.

Chicago's development investments for mobile computing have focused on delivering solutions for these two problems. More specifically, the development focus has been on the following:

- Keeping mobile PC users connected to and communicating with people both within and outside of their workgroup (LAN).
- Seamlessly adapting to hardware changes when users demand different hardware as they move from location to location, task to task.

The following table shows features meant to address the communication and hardware adaptation needs of users:

	Communication	**Hardware Adaptation**
Empowering	Remote Access	Warm Docking
Users	The Briefcase: File Synchronization	Power Management
	Local Connections: Serial, Parallel,	PCMCIA Support

	Wireless	
	Remote Mail	Dynamic Networking
	At Work Fax	Deferred Printing
	Accessing Internet and Online Services	Dynamic Video Resolution
	VoiceView	Enhanced Mouse Support
	User Profiles	
	Password Management	
	Document Viewers	
Enabling Independent Software Vendors (ISVs)	Chicago Networking Architecture	Plug and Play Architecture
	Slow Link and Remote Access Services	Device Detection and Setup
	OLE 2: File Synchronization Interfaces	Plug and Play messages
	Telephony API (TAPI)	Enhanced Registry
	Unimodem	
	Messaging API (MAPI)	

The following sections detail how Chicago provides improved communication and hardware-adaptability features for the mobile user.

Remote Network Access

In the office, well over 50% of PC users have become accustomed to full workgroup computing capabilities—printing to a network printer, sending and receiving email, and accessing shared files. However, when users leave the office, while they can take their computer home, they cannot take all of the shared resources from their workgroup environment with them.

Chicago's remote access feature gives users complete workgroup computing capabilities whether it be running a client-server application accessing a customer database, downloading and/or browsing email, or accessing shared files. It is smoothly integrated into the Chicago shell so it doesn't seem like users are running a separate application. Establishing a remote connection works the same as establishing a connection in the office—the user simply double-clicks on the network object to connect. Similarly, if the user double-clicks on Mail, a remote connection is automatically established.

The Remote Access client software component, like the rest of Chicago networking, connects to a broad set of networks and provides an open architecture for universal connectivity. Chicago's remote access functionality includes support for IPX, NetBEUI, and TCP/IP network protocols using industry standard point-to-point protocol (PPP) over the wire, as shown in Figure 73. Because remote access support uses the dynamic, 32-bit, protected-mode network architecture of Chicago, users don't have to reboot their computers or exit Windows to continue working after establishing or ending a connection.

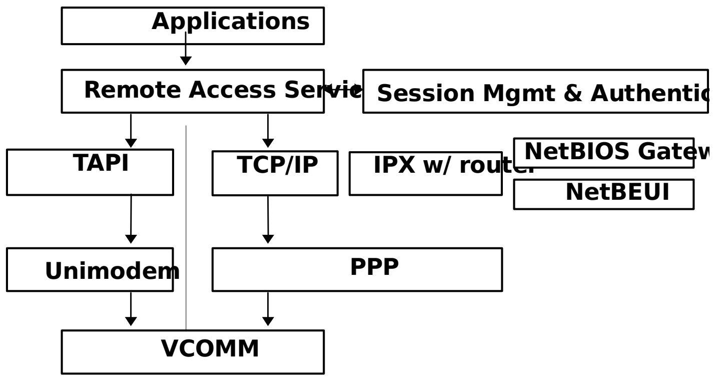


Figure 73. Remote Access Functionality in Chicago Supports TCP/IP and IPX

On the host computer—that is, the one into which the user is dialing—Chicago provides an easy to use, single-port host, capable of multi-protocol routing for IPX and NetBIOS with pass-through user-level security. This security uses Windows NT or NetWare authentication mechanism and user database to validate the user. Of course, share-level security is also an available option. Using Chicago's desktop management capabilities, an administrator can disable dial-in access so users cannot dial into a particular desktop PC, or remotely access the entire network. (For more details about Chicago's desktop management infrastructure, see the Networking section earlier in this guide.) If the user chooses to use a host system such as Windows NT, Shiva Netmodem/LanRover, or NetWare Connect, Chicago's remote access feature offers full connectivity. This also means that the Chicago desktop PC can be used as a convenient access point to a small LAN or simply to the desktop PC itself. (The "Daytona" release of Windows NT Advanced Server supplements the remote network access support in Chicago to provide a large network solution that allows for as many as 256 simultaneous dial-in sessions.)

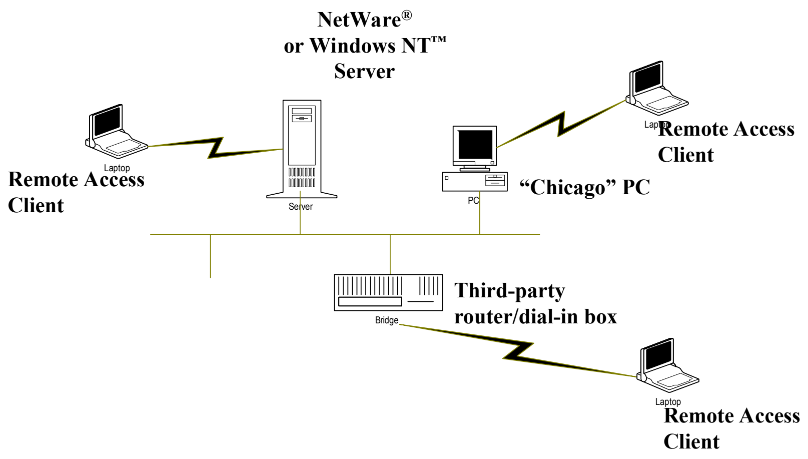


Figure 74. Chicago Supports Flexible Remote Connectivity Options and Broad Network Access

Chicago provides a modular, open architecture to support applications establishing a "pipeline" to the network remotely. Part of the Win32 API, is the Remote Access API, which provides ISVs with services to initiate resume a remote connection as well as gather information about the type and status of the connection.

The second critical element of these services is the Remote Access subsystem. This open subsystem is network- and device-independent to enable universal connectivity. This means, for example, that Chicago supports ISDN boards, PBX modems, and so on. This capability is accomplished through *service providers*—software components that manage physical connections, and network traffic over the remote media. Finally, the Remote Access subsystem has a modular authentication provider that can be supplemented or replaced to provide custom security services. For instance, suppose the BrandX Company wanted to provide its own custom services. That company can replace Chicago's authentication DLL with its own to take advantage of BrandX-specific security features.

The following diagram highlights the various components of the remote access subsystem that can be replaced by third-party service providers. The shadowed items can be replaced to add functionality not offered by the basic provider.

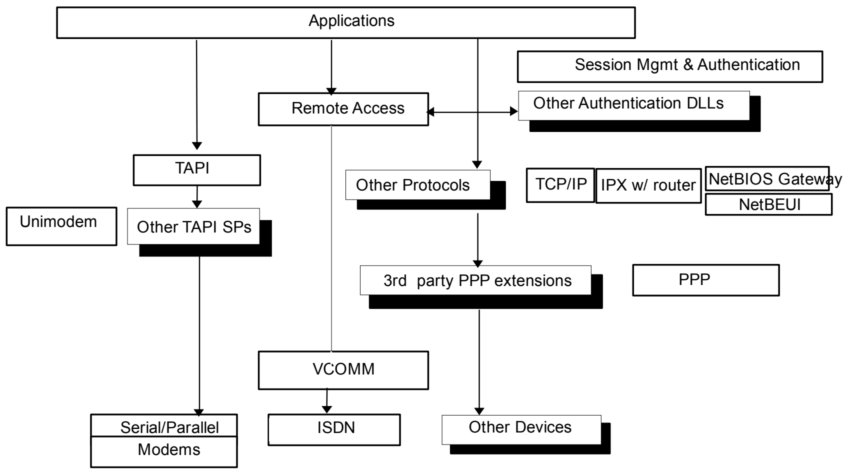


Figure 75. Authentication DLL Integrates Directly Into PPP Connectivity

Telephony API

To communicate in a mobile environment, users and applications must dial phones or modems. Applications can use Chicago's TAPI to dial, whether the device is a phone on a PBX system, an ISDN board, or a modem. TAPI provides services to allow applications to share a line so that more than one application can wait for an incoming call while another dials out. TAPI itself is extensible so third-party developers can write TAPI service providers to extend the dial support of TAPI. One such TAPI service provider is Unimodem (discussed in detail in the following section).

TAPI also provides Dial Guide (this is the preliminary name for this user interface that applications can call) to guide users through the process of defining the correct phone number. Dial Guide gives users the opportunity to define phone numbers in a location independent fashion. The user enters an area code and seven digits out of their Address Book, for example, and using Dial Guide applies location specific parameters to the number, such as a prefix to get an outside line. When users dial this number from different location, they simply switch their Dial Guide locations, instead of reentering the entire number.

Unimodem

Similar to Window's infrastructure for printers, Chicago provides an easy, central, extensible mechanism for installing and configuring modems. It is important to note that TAPI and Unimodem use Chicago's extensible 32-bit communications architecture. See the Communications Architecture section of this guide for additional information. Chicago automatically detects the modem and provide a default configuration for it. Once the user installs a modem, it is available to all applications. Applications don't have to store modem commands or the capabilities of the different modems. Chicago ships with support for the top 200 modems worldwide. Adding new modems is as simple as supplying the appropriate .INF file. (Microsoft certifies the .INF files for each new modem and the modem includes a logo identifying it as Windows compatible.)

“Local” Connections

Roughly 70% of portable PC users also own a desktop PC. As a result, they constantly need to connect their desktop and portable PCs “locally” (that is, from a range of one to ten feet.). Chicago makes this process significantly easier.

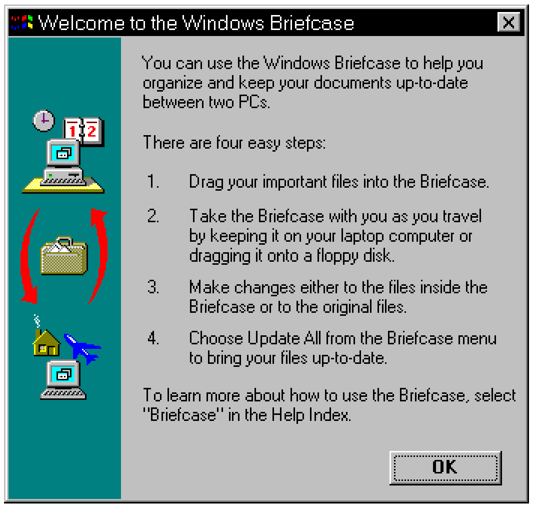
Like remote access, establishing a local connection is seamlessly integrated into the shell and provides full participation on a variety of networks.

Wireless technologies like Infrared (IR) are another form of local connections. Using Chicago’s extensible device driver architecture, Microsoft is working closely with creators of wireless devices to develop and ship Windows drivers for these new technologies.

File Synchronization: The Chicago Briefcase

Portable PC owners who also have a desktop PC constantly work to keep the most up-to-date files on the computer they are currently using.

The Chicago Briefcase minimizes these headaches by keeping track of the relationships between the files on their two computers using a simple metaphor that users are already comfortable with—a briefcase.



You can use the Windows Briefcase to help you organize and keep your documents up-to-date between two PCs.

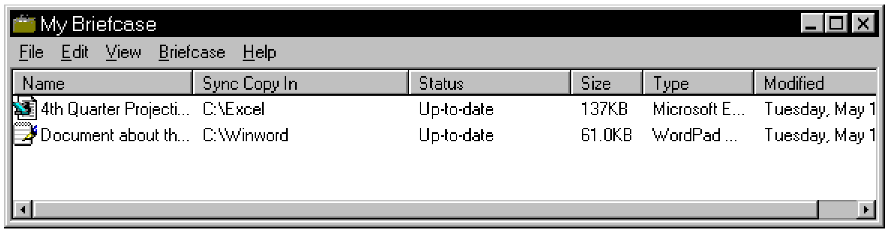
There are four easy steps:

1. Drag your important files into the Briefcase.
2. Take the Briefcase with you as you travel by keeping it on your laptop computer or dragging it onto a floppy disk.
3. Make changes either to the files inside the Briefcase or to the original files.
4. Choose Update All from the Briefcase menu to bring your files up-to-date.

To learn more about how to use the Briefcase, select "Briefcase" in the Help Index.

Figure 76. Initial Briefcase Dialog Box Explaining Briefcase Process

After installing the Chicago Briefcase software on the portable PCs, a user can specify which files and directories he or she wants to take keep up-to-date between the two PCs by dragging and dropping those objects into the Briefcase. When the user reconnects the portable and desktop PCs, Briefcase automatically replaces the unmodified files with the recently modified files. If both files have changed, Chicago calls the appropriate application to merge the disparate files.



Name	Sync Copy In	Status	Size	Type	Modified
4th Quarter Projecti...	C:\Excel	Up-to-date	137KB	Microsoft E...	Tuesday, May 1
Document about th...	C:\Winword	Up-to-date	61.0KB	WordPad ...	Tuesday, May 1

Figure 77. Sample Briefcase Contents Showing Document Status

Chicago provides a set of OLE 2 interfaces that allow applications to bind reconciliation handlers to it, track the contents of the Briefcase, and define the outcome of any reconciliation on a class by class basis. For example, when both the file in the Briefcase and its synched copy outside have changed, Chicago calls the appropriate reconciliation handler to merge the two files. These APIs will also serve as the foundation for Cairo's reconciliation APIs. As a result, ISVs writing to the Chicago reconciliation APIs can leverage that investment as they write Cairo applications in the future.

Remote Mail

Historically, a user leaving the office also left behind email capabilities. Microsoft, Lotus and other email vendors are changing this. Chicago delivers the next generation of remote mail. With Chicago, mail is completely integrated mail into the Chicago shell so that a user's Inbox appears as just another folder rather than a separate application. When out of the office, users don't need to alter their behavior. They simply connect a phone line to their modem and start using mail. The remote connection is established automatically for them using Remote Access services.

While connecting remotely is seamless to the user, Chicago has minimized the cost of slow links for the user. Performance over the wire has been enhanced, users can browse headers, downloading only specific messages and getting an estimated time to download and status of the download process.

Messaging API

MAPI provides a critical layer for Chicago's mobile story. Mobile users more than any other class of users need multiple messaging providers and the ability to seamlessly move between these providers.

MAPI is an open, extensible messaging infrastructure standard. This standard ensures complete independence for Windows applications and client software from underlying messaging systems while enabling vendors to supply a wide array of providers. MAPI provides the support to dynamically switch in and out providers and associate multiple providers and preferences with a "profile."

Microsoft At Work Fax

Fax is one of the primary means in which mobile users send messages and documents. Rich fax services are seamlessly integrated into the Chicago email client. Users send a fax the same way they send any other electronic message.

For example, these services enable users to send a fax and attach a binary file just the way they do in mail today. Depending on the capabilities of the recipient's PC or fax machine, the message appears as a message in their inbox with an attachment or for those people with a Class 3 fax machine, the attached document is rendered and printed with a cover sheet. If users want to send faxes when they are not connected, they can spool them to their outbox and when they connect, the faxes are automatically be sent. Microsoft At Work Fax provides security to ensure the correct recipient via an RC4 encrypted password or public key and private key encryption.

Microsoft At Work Fax uses the open, extensible architecture of MAPI, plugging in as a transport provider and then leveraging the user interface provided by the Chicago client.

Accessing Online Services

Online services are a critical network for PC users to gather information and communicate with other PC users.

Chicago integrates a variety of access points into online services. The Info Center will allow Chicago users to send and receive mail through a variety of mail service providers from a single integrated mail client. Chicago's HyperTerminal communications application will support dial-up connections to services including CompuServe, Dow Jones, and America Online.

User Profiles

Many users share portable and desktop PCs, but each user wants individualized settings for user interface appearance and operating system defaults. Ideally, as users move from one PC to another, these settings move with them. This is possible with Chicago's user profiles.

A *user profile* is a list of user-specific settings, kept on a per-user basis, in the HK_Current_User key of the Registry. Because user profiles are per-user, even users who share a computer can each maintain separate user profile of customized settings for Chicago.

User profiles can also be stored up on the network to address the needs of the roving PC user. When the user logs onto Chicago (and thus the network) on a *different* PC, their settings can be downloaded from the network.

Password Management

Users constantly strive to protect the data on their portables from prying eyes and hands. This is no easy chore. Password protection at boot-up, after a suspend (reduced power) state, and at network logon means users must often contend with scattered user interface and multiple passwords.

Chicago's Security Control Panel provides a central, extensible mechanism for users to manage their security. Chicago's Master Password gives users the opportunity to unify all their different passwords around a single password regardless of the physical store of the password. This user interface and security are open and extensible. As a result, ISVs and portable PC manufacturers can add their own property sheets to the Security Control Panel and hook their password services to the Master Password.

Document Viewers

Portable PC users often need to exchange and view documents with customers or people in a different work environment then their own. Many times, however, they don't have the application used to create that file so they can't view the file..

Seamlessly integrated into the Chicago shell is an extensible, replaceable file viewer technology. Users simply select a file and choose "Quick View" to view the file. Chicago supports viewing more than 30 different file types out of the box and publishes interfaces to allow applications to add support for additional formats and even to add their own viewer.

Docking Support for Mobile Computing Hardware

Many portable PC users have had to compromise storage, extensibility, and display size and resolution in favor of mobility. Docking stations provide users with both the mobility of the portable PC and the storage, extensibility, and versatile display capabilities of a desktop PC. However, users with docking stations spend a lot of time synchronizing files on the desktop and portable computers.

Chicago makes this kind of synchronization automatic for the user.

Microsoft forged partnerships with leading portable vendors like Toshiba and Compaq and BIOS vendors like Phoenix Technologies to achieve a level of integration between hardware and software never achieved before. On the hardware side, the challenges were to electrically enable docking without powering off the computer. On the software side, Chicago must detect the changes in configuration anticipate the resulting changes in hardware, manage any conflicts, and load the appropriate drivers.

The results are exciting. Chicago and an accompanying family of hardware support docking without powering the computer off. Chicago ships with a full complement of dynamically loadable drivers from network drivers to mouse drivers that load and unload depending on whether the user is docked or undocked.

To the user, they simply choose to “Eject PC” from the System menu. Chicago checks for any potential problems before undocking (for example, Chicago lets a user know if he or she has a file open on a network drive at this time), and the system undocks, without powering down if the user chooses. Once undocked, the system automatically reconfigures for the different hardware (for example, changed video resolution down to 640x480).

New Message Support

The Windows Plug and Play initiative provides is a new set of Windows messages that alert applications and device drivers to changes in the hardware so they can react intelligently. These messages include:

- **Docking:**
  About to change configuration (for example, when the user is about to undock)
  Device about to be removed
  Configuration changed (for example, when the user just undocked)
  Device about to be added
- **Power Management:**
  System about to suspend
  System suspended
  System resumed
- **PCMCIA:**
  Device inserted
  Device removed
- **Miscellaneous:**
  New device inserted (for a device that needs to be set up)
  Serial mouse inserted
  Parallel cable inserted

These messages expose a plethora of opportunities for applications and system services to better support the portable PC user. Chicago itself takes full advantage of these messages. Take the Config_Changed message as an example. Applications shipping with Chicago alone uses the message this way: the Briefcase uses it to try to start updating, the print spooler to print all deferred print jobs, and Mail to try to reestablish a network connection.

The Registry

The Chicago Registry provides a centralized, dynamic data store for all Windows settings that ISVs can use. The Chicago Registry defines a branch called HKCurrent_Config to better serve the needs of a mobile user. The Config branch stores information on a per-configuration basis. For example, the Desktop Control Panel stores per-configuration information about video resolution changes and Print Manager stores per-configuration information about the default printer.

Configurations are created when Chicago queries the BIOS for a dock serial ID, asking the user for a friendly name for the configuration, and then storing hardware and software associated with this configuration. Applications access and store information for each of the different hardware configurations that the mobile user uses.

PCMCIA Support

The emergence of PCMCIA cards has been one of the most exciting advances in the portable market. However, to date, users were never sure if the card was compatible with their portable, they had to struggle through installation, and insertion and removal was anything but dynamic.

Through the Plug and Play architecture, Chicago's PCMCIA support brings power, compatibility, ease of installation, and dynamic card insertion and removal to the PCMCIA experience. The drivers are robust, 32 bit, dynamically loadable virtual device with zero-memory footprint. Chicago ships with an updated version of card and socket services and a compatibility testing/logo program ensures compatibility with these standards. Installation is as simple as inserting the card. Finally, insertion and removal of cards happens dynamically. For example, when the user plugs in a PCMCIA network card instead of having to reboot the computer, the portable detects the network card, load the network stack, establish a network connection (assuming one is available). Then the shell updates its user interface to reflect that the mapped network drives are now active.

Dynamic Networking

Historically, network users with a portable have dealt with CONFIG.SYS files and a regular stream of error messages as they connected and disconnected from the network.

To adapt to changes in link speed and configuration, Chicago's network solution is *completely* dynamic, regardless of whether the user is using the NetWare compatible components, or a Windows networking components. All the underlying drivers, transports, and redirectors are robust, 32 bit, dynamically-loadable, protected-mode virtual devices that support Plug and Play. This enables Chicago to load and unload components of the network stack as demanded by hardware events. For example, when the user docks a portable PC, the network components are loaded and connections are established without user interaction. This is seamless mobility. Even assigning a TCP/IP address is now dynamic, using the Dynamic Host Configuration Protocol (DHCP) servers to allocate addresses on demand.

Power Management

One of the curses of a portable user's existence is battery life. Chicago supports Advanced Power Management (APM) 1.1 which represents a major step forward from APM 1.0.

From an end-user perspective, there are three major changes. First, integrated into the Chicago shell is a battery meter that lets the user constantly monitor an accurate representation of the battery life they have remaining. Second, from the System menu, the user will be able to put their system in "Suspend" as opposed to going to a hardware control. Third, users will have the option of automatically powering their PC off when they shut down Windows. (In the past of course, the user had to shut down Windows, then use the hardware power switch to shut off the PC.)

From a software vendor's perspective, Plug and Play APM messages allow applications to react to changes in the power state and battery life.

Flexible Video Resolution Support

In one focus group after another, display was cited as the number one limitation with portables. To overcome that limitation, portable vendors are putting high-end video on the motherboard, and users are plugging a monitor to their portable when they are at their desks.

Chicago stores video resolution on a per configuration basis and supports dynamic resolution changes. As a result, when the user is docked, for example, they can set their video resolution at 1024x768. Whenever they return to their docked configuration, the resolution will automatically return to 1024x768.

Deferred Printing

Users want to generate print jobs regardless of where they are. Chicago supports deferred print jobs so the user can generate print jobs regardless of whether a printer is connected. When a printer becomes available, Chicago detects this connection and automatically spool the print jobs as a background process.

Extending this, Chicago gives users the ability to print to a generic printer. So if they aren't sure which printer they will be connected to next, users can still queue the print jobs, then specify the printer only when they connect to the printer.

Finally to better support the mobile user, Chicago stores the default printers on a per configuration basis. If the user has a different printer at home then they do the office, when Chicago detects the change in location (for example, docked versus undocked state on a computer), it will change the default printer.

Pointing Devices

In our focus groups, portable PC users often described difficulties in switching between the integrated pointing device on their portable to a desktop pointing device when they are at their desk.

Chicago improves on this in two ways. First, when the user changes configurations, Chicago automatically detects which pointing device is available to use, and enables it (or both). Extending this, without changing configuration, users can plug in a Plug and Play serial mouse and the system will detect the new mouse, and dynamically reconfigure so the user can it.

Chicago’s Messaging and Information Center

A Single View on a World of Information

Customers today are using their PCs for an increasingly wide range of tasks, beyond simply creating and editing documents. Electronic mail has not only become a primary communication vehicle within many organizations, but also among companies, organizations, and individuals. Many companies and departments have begun installing so-called workgroup systems, which provide team information sharing capabilities beyond basic email. Finally, the world of online information services has begun to dramatically increase—witness the astounding 15% per month growth rate seen by the Internet, in addition to the rapid growth in commercial services, such as CompuServe® and others.

A problem users face today is that each of these different information sources and services comes with its own unique software and user interface. Users often have software for an email client such as Microsoft Mail®, a groupware client such as Lotus Notes®, and an online services client such as CompuServe Information Manager®, and perhaps some Electronic Fax software they received with their modem—all in addition to the basic File Manager they use for accessing and manipulating documents. Users with email accounts on multiple services face an even more complicated problem, having to log into to several different email applications simply to check their mail.

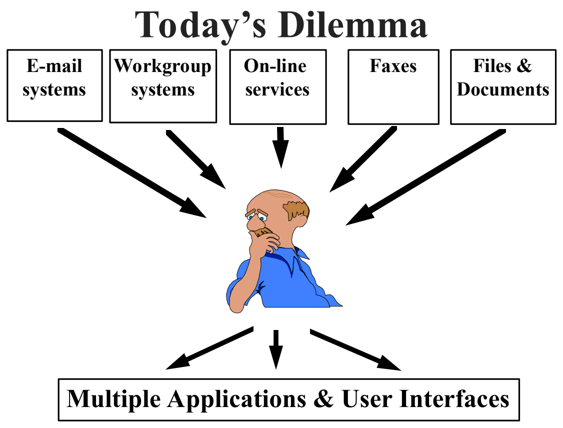


Figure 78. Today's Dilemma Involves Integrating Disparate Information Sources

To work with different types of information, the user today must acquire and learn four, five, or even more separate applications. Often there’s very little integration between programs beyond rudimentary cut and paste.

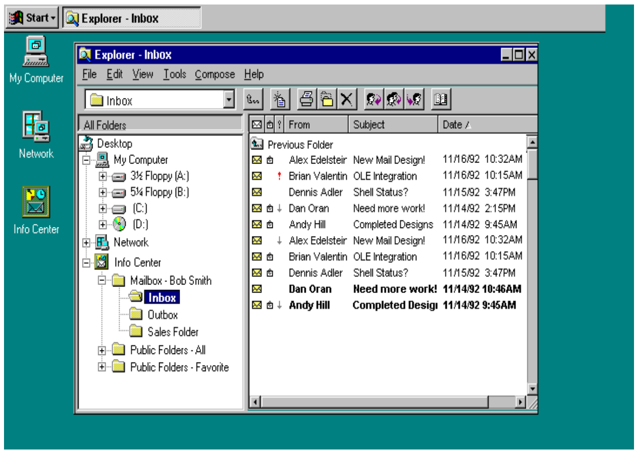
To make using the PC interface easier, Chicago consolidates the previously separate applications functionality provided by File Manager, Program Manager, and Print Manager into a single UI to provide common and consistent access. Chicago does this by extending the Windows shell beyond just files and documents in the users' local and network file system. Users now have a single tool—Windows—that serves as a platform that provides easy, common access to a wide variety of email systems, workgroup applications, and information services.

The Info Center

As described earlier, the Chicago shell features a single namespace that represents all of the data accessible locally and on the network. Through the new user interface, new users are protected from the complexities of a large hierarchy of information by Chicago's simple folder metaphor. Advanced users can choose to make full use of this powerful namespace by using the Explorer tool to see the entire hierarchy at once. The Chicago namespace is initially divided into three fundamental "places":

- **My Computer**—all of the files, documents stored physically on the user's PC.
- **Network Neighborhood**—all of the files and documents that reside on other PCs on the network.
- **Info Center**—email messages, forms, documents, and folders that have been stored by some information service, such as an email system or workgroup application. Items in the Info Center needn't be MS-DOS files on a hard disk—the actual storage mechanism can be an email system or a workgroup database server.

Messaging and information services "plug in" to the Info Center. It provides a common interface in Windows where users access all of the services they use. Users can choose to view the Info Center resources as regular folders (the default), or they may "explore" the Info Center. The figure below shows the Info Center opened with the Explorer, in this case open to the user's Inbox folder.



From	Subject	Date
Previous Folder		
Alex Edelsteir	New Mail Design!	11/16/92 10:32AM
Brian Valentin	OLE Integration	11/16/92 10:15AM
Dennis Adler	Shell Status?	11/15/92 3:47PM
Dan Oran	Need more work!	11/14/92 2:15PM
Andy Hill	Completed Designs	11/14/92 9:45AM
Alex Edelsteir	New Mail Design!	11/16/92 10:32AM
Brian Valentin	OLE Integration	11/16/92 10:15AM
Dennis Adler	Shell Status?	11/15/92 3:47PM
Dan Oran	**Need more work!**	**11/14/92 10:46AM**
Andy Hill	**Completed Desig**	**11/14/92 9:45AM**

Figure 79. Chicago Desktop Showing Info Center

Within the Info Center, the user can:

- **Read and Send Email** on a number of different email systems, while maintaining a single inbox for all incoming messages. Drivers will be available from a wide range of vendors, including LAN-based systems, host-based systems, and various online information services.

- **Send and Receive Faxes** electronically with the use of a fax modem supported by the Microsoft At Work Fax software. Faxes are sent just like any email message, and incoming faxes are received in the same inbox with other email.

- **Access Workgroup Servers** including email, shared databases, forms packages, document stores, and workflow systems with the use of appropriate MAPI drivers.

- **Easily Move Messages and Documents** back and forth between folders. Users can store information the way they want to—email messages can be mixed with word-processing documents and spreadsheets. The Windows shell provides a common user interface and toolset for managing these different types of information.

- **Organize All Types of Information** for most effective retrieval and use. Folders stored under the Info Center allow the user to create customized views of their information—users can choose which fields (or "properties") to display in these folders, and how to sort and filter the stored items.

Info Center Components

Chicago's Info Center system is the part of the Windows operating system that allows Windows to handle all these different types of information. It consists of a number of components:

- **A Set Of Basic MAPI Drivers**. Chicago includes MAPI drivers for Microsoft Mail as well as Microsoft at Work Fax services. Additional drivers will be available from 3rd parties.
- **The Info Center "Viewer"**. This is an advanced, extensible messaging and workgroup client that is built into the Chicago shell. It can be the front-end to any email or information system that has appropriate MAPI drivers - including Microsoft Mail. It can be easily customized to display advanced features when connected to an advanced messaging system. It includes an OLE 2.0-compatible rich text editor used for composing and reading messages, as well as powerful custom views, searching, and filtering.
- **Common Address Book**. The Windows Address Book contains not only email addresses, but names, phone/fax numbers, mailing addresses, and other personal contact information. Through the open MAPI interfaces, it is accessible from a wide variety of applications. Through the use of MAPI drivers , the Address Book is also the user interface for corporate email and information services directories. The Windows Address Book can store addresses for multiple email systems at the same time.
- **Personal Information Store**. The Windows Messaging System includes a sophisticated local "database" that allows users to store messages, forms, documents, and other information in a common place. Its rich organizing capabilities include using long filenames, plus sorting and filtering on various fields of the stored objects. Users can create and save any number of custom views on the information in their personal store. It also functions as the user's Mailbox—including a universal Inbox and Outbox that work with any connected messaging system.
- **MAPI 1.0**. The core system components that connect the Viewer and other applications to the various information services. MAPI's namesake component is the Messaging Application Programming Interface — the set of services that any mail-enabled or workgroup application can make use of. MAPI also defines a Service Provider Interface (SPI) that allows drivers (or "providers") to be written for many different messaging and workgroup services.

Open Architecture for Open Connectivity

The Info Center is designed to work with virtually any messaging or workgroup system—whether it's LAN-based, host-based, or an online service requiring dial-in access. The key to this open architecture is MAPI—sometimes called the "Windows Messaging System."

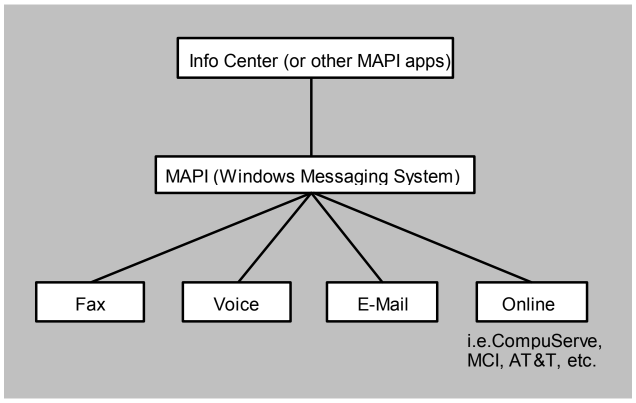

Info Center (or other MAPI apps)
MAPI (Windows Messaging System)
Fax
Voice
E-Mail
Online
i.e.CompuServe, MCI, AT&T, etc.

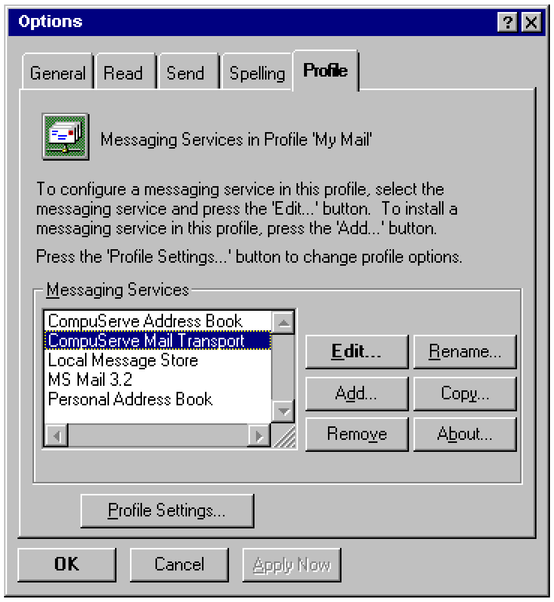
Figure SEQ Figure * ARABIC 80. MAPI Provides an Open Architecture

Any information or messaging service provider can develop a MAPI "driver" (see below) so that the Info Center can be used with their system. There are currently over 60 different vendors working on creating MAPI drivers for different email, voicemail, workgroup, and information systems—including LAN-based, Host-based, and Wireless systems. A partial list of these providers includes:

- Apple
- AT&T
- Banyan
- CompuServe
- Delrina
- Isocor
- Lotus
- MCI
- Microsoft
- Novell
- Octel
- RAM Mobile Data
- Skytel
- WordPerfect

Profiles

A Chicago user can install a combination of drivers so that their Info Center can be used for multiple email or workgroup systems at the same time. To make it easy to use different information services, Chicago allows users to set up profiles. A user's profile specifies which messaging and information services the user will have access to, along with preference information and settings. In the example below, the user has set up a profile that contains both CompuServe Mail as well as Microsoft Mail. This user will be able to send and receive on both systems simultaneously—while sharing a common inbox and address book. Profiles are stored in the Chicago Registry on a per-user basis.


Options

Messaging Services in Profile 'My Mail'

To configure a messaging service in this profile, select the messaging service and press the 'Edit...' button. To install a messaging service in this profile, press the 'Add...' button.

Press the 'Profile Settings...' button to change profile options.

Messaging Services

CompuServe Address Book
CompuServe Mail Transport
Local Message Store
MS Mail 3.2
Personal Address Book

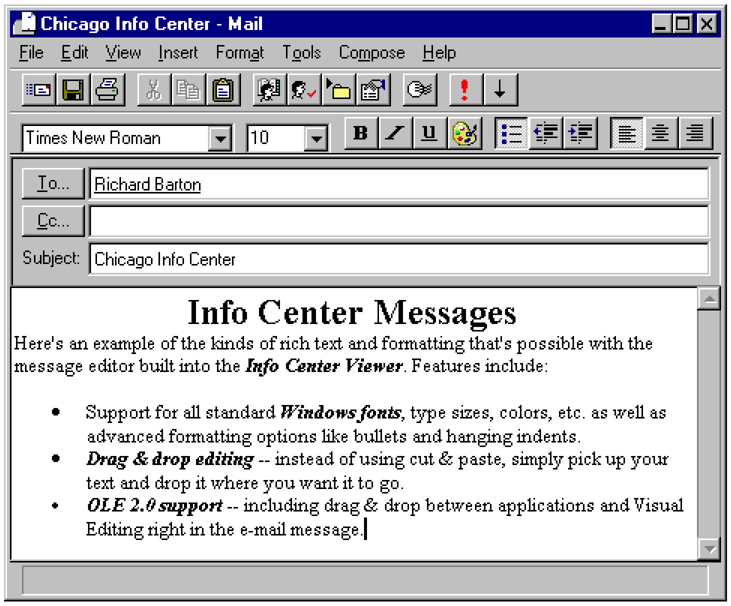
Figure 80. Configuring a Profile with Multiple Services

A Quick Tour

To best understand the role the Info Center would play in a user's day-to-day access to electronic mail and online services, a sample scenario is presented as follows.

To read their email in Chicago, a user would first log onto Windows. With many of the MAPI drivers under no separate logon is required for the mail system—it has been unified with the "system" logon. For simple access to the user's common tasks, Chicago has a Start button. One of the items on the Start button is "View Inbox."

This immediately takes the user to the inbox (see Figure 79 above) where he or she can read any new mail received. Users can compose new messages or choose to reply to existing ones. There is also a "Compose New Message" item on the Start Menu. The Info Center includes an OLE 2.0-compatible rich text message editor that lets the user get the point across in an effective way using a combination of fonts, font styles, and font attributes.


Chicago Info Center - Mail
File Edit View Insert Format Tools Compose Help
Times New Roman
10
B I U
To...
Cc...

Richard Barton

Subject: Chicago Info Center

Info Center Messages

Here's an example of the kinds of rich text and formatting that's possible with the message editor built into the ***Info Center Viewer***. Features include:

- Support for all standard ***Windows fonts***, type sizes, colors, etc. as well as advanced formatting options like bullets and hanging indents.
- ***Drag & drop editing*** -- instead of using cut & paste, simply pick up your text and drop it where you want it to go.
- ***OLE 2.0 support*** -- including drag & drop between applications and Visual Editing right in the e-mail message.

Figure 81. Rich Text Message Editor

Note that this same message editor can be used with any of the back-end messaging systems that have MAPI drivers available. If the underlying messaging system doesn't support rich text, the message can be sent as plain text to maintain compatibility. Or, if the destination reader also uses the Info Center Viewer the rich text information can be automatically encapsulated and sent as a binary file.

The user can address this message to any user on any of the email systems to which he or she is connected. Additionally, the message may be sent as a Fax by choosing a recipient from their address book who has a Fax address, or simply entering [FAX:*phone#*] on the TO: line.

Messages received in the inbox can be saved for future reference, if the user so chooses. The user simply drags the messages into any of the other folders in the mailbox (message stores)—or the user can drag the message to any folder on their local or network hard drives. In the latter case, the message becomes a .MSG file—but maintains all of the messaging-specific fields such as Sender, Recipient, and so on.

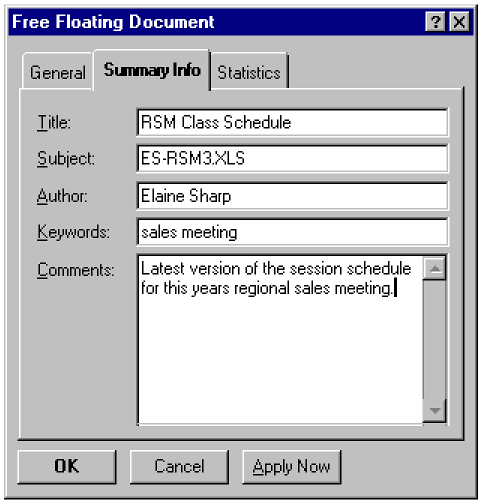
Information Stores

Sets of folders in the Info Center are also called Information Stores. Users can also drag items from the file system into information store folders. Information Stores go beyond the basic MS-DOS file system in many ways. They can be physically stored in local files, or represent a database on a network server. One of the differences between the file system and an information store is the set of fields -- called *object properties* -- that are stored along with each item. Object properties include not only messaging-related properties like Sender, Recipients, Date Received, Subject, and so on—but also custom properties including:

- **Fields Defined In Custom Forms.** MAPI provides an open method for forms developers to register their forms and related data fields. Once registered, these fields become object properties suitable for searching, categorizing, and so on. This facility is available to any forms package or development tool.

- **Standard OLE 2.0 Document Properties.** Applications that are compatible with the OLE 2.0 object standard can choose to store their documents in a standard compound storage format. Compound documents of this type have a number of pre-defined standard properties, such as Author, Title, Keywords, Comments, and Number of Pages. Users can simply drag a compound document into an information store, and the document properties are automatically extracted so they can be used in views, searching, and so on.

  The figure below shows the OLE properties of a compound document that has been dragged into a folder in the user's information store:



Title: RSM Class Schedule
Subject: ES-RSM3.XLS
Author: Elaine Sharp
Keywords: sales meeting
Comments: Latest version of the session schedule for this years regional sales meeting.

Figure 82. OLE Document Properties

- **Custom OLE 2.0 Document Properties.** Many applications allow their users to define custom properties; for example, Case Number for a law firm or Total for an expense report. Like the pre-defined properties above, these will be available in the views, filters, searching, and so on.

The data in a Personal Information Store is kept in a single file, making it easy for companies to distribute rich documents and messages in a standard format. Every Chicago user then has a built-in tool (Info Center Viewer) that can browse, search, and organize the information in rich ways. Personal Information Stores can be encrypted and password-protected for security. Besides the Personal Information Store, the MAPI architecture makes it possible to plug in many different types of servers and databases as information stores.

The Info Center - Summary

The Info Center architecture then, provides Chicago with a single toolset and user interface for accessing, exchanging, and organizing information—regardless of data type, including E-mail, faxes, documents, and others. Through MAPI, it provides an extremely open platform—making the Windows desktop the “place to live” regardless of the back-end services in use.

Multimedia Services

For the past year, the home market has been the fastest-growing segment of the PC business, and multimedia titles have been one of the fastest-growing segments of the software industry. A large and increasing portion of the PCs being sold into homes are coming with the equipment that makes cool multimedia applications possible − notably CD-ROM drives, sound subsystems, horsepower, and local-bus video.

In 1993, the installed base of multimedia-capable Windows PCs grew rapidly to become the largest multimedia computing platform in the world.

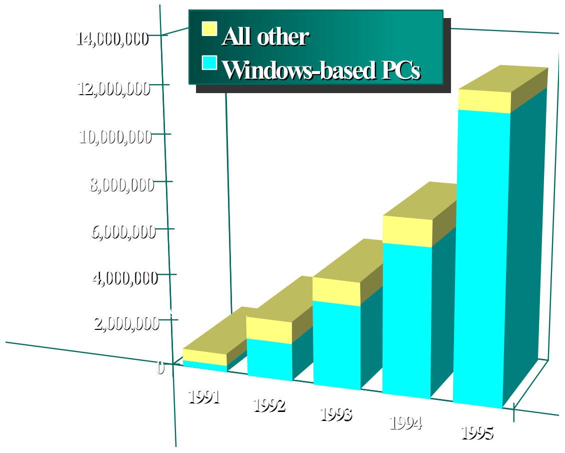


Figure 83. Estimated and forecast sales of multimedia-capable PCs. **Source: Dataquest**

By Christmas of 1993, there were more multimedia titles available for Windows than there were for any other computing platform.

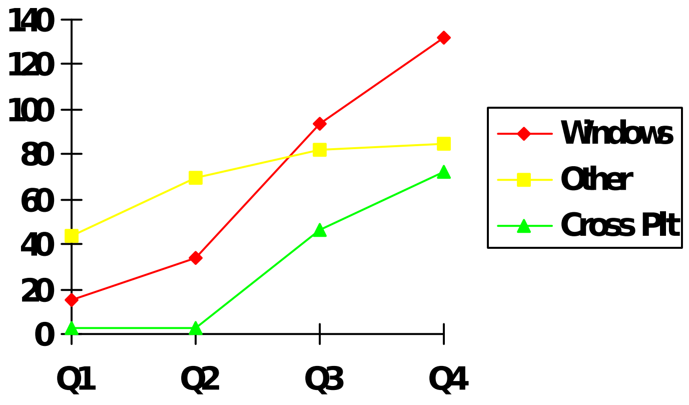


Figure 84. Number of multimedia titles sold by computer software retailers in 1993, by platform, by quarter, as reported by PC Data.

Microsoft Windows and Multimedia

Microsoft is committed to making Windows the leading force in multimedia technologies and systems for personal computers. Our commitment takes many forms, but the most important one is our ongoing investment in multimedia-related research and development. Some of the end results of the last few years of research and development are described in this chapter. This is far from the end, though. Multimedia technologies are evolving rapidly, and we will continue to press ahead in providing tools and architectural enhancements to enable developers and consumers to take advantage of new innovations.

A Little History

It is worth dwelling for a moment on how far Windows multimedia has come in the last few years. When Video for Windows 1.0 was released in 1992, sound cards and CD-ROM drives were relatively rare. Graphics subsystems were universally ISA-based, and software codec technology was in its infancy. The standard size for a digital video clip in this timeframe was 160 pixels by 120 pixels − one-sixteenth of a VGA resolution screen. Technologists (who understood how difficult this was to accomplish) cheered wildly and proclaimed the dawn of the multimedia computing era. Customers shrugged. What's so great about a video clip the size of a dancing postage stamp?

In 1993, hardware and software makers began to deliver equipment and technology that offered better-than-postage stamp performance at reasonable consumer prices. Double-speed CD-ROM drives and local bus video offered more bandwidth to support the massive data requirements of digital video and quality sound. A second generation of software codecs made more effective use of the data available. Prices on 16-bit sound cards dropped into consumer range. With Microsoft Video for Windows 1.1, the size of a digital video clip that a mainstream computer could display reliably increased to 320 x 240 − one-quarter the size of the screen. On one hand, critics have a point when they label this sort of digital video "dancing credit cards". On the other hand, digital video of this size has proven compelling enough to consumers that it has spurred a virtual tidal wave of multimedia title development. Retail software store shelves are crowded with multimedia titles and games.

Progress marches on. Installing Chicago will provide today's multimedia PCs with an overnight upgrade in multimedia capabilities. Based on the capabilities of high-end PCs in 1994, the mainstream PC of 1995 will be able play digital video segments that are larger, smoother, and better-looking than ever before − even up to 640 x 480 (full screen) and beyond. We are now able to look forward quite realistically to a time when the amount of data that can be stored on a CD-ROM (rather than the speed of the video subsystem) is the most relevant factor limiting the richness of a consumer's experience with a multimedia title or game.

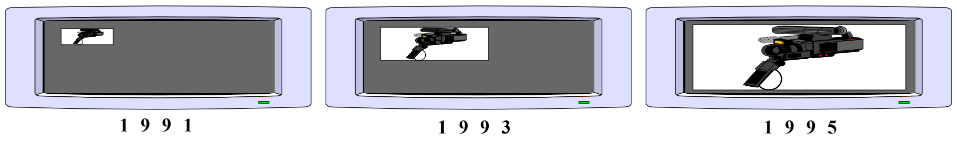


Figure 85. The dancing postage stamp era of multimedia computing is over.

Chicago—A New High-Performance Multimedia Platform

Chicago delivers a new high-performance platform for PC multimedia. From a Big Picture perspective, here's the "greatest hits" of what Chicago contributes to the world of multimedia computing:

For consumers, Chicago makes multimedia easier, more fun, and more engaging.

- **Easier.** Plug and play will make it far easier for consumers to install multimedia devices successfully. All of the architectural support for digital video, audio and MIDI is built into Chicago, so that users are liberated from setup challenges. And Chicago is compatible with multimedia titles and tools created for Windows 3.1.

- **More Fun.** Chicago is a much better platform for computer games than any version of Windows has ever been, including support for fast, intensely graphical games.
- **More Engaging.** Installing Chicago is an immediate multimedia upgrade that allows any PC to become a better, more exciting multimedia playback machine. Authors creating titles and games for Chicago will be able to make their products faster and more exciting to play.

For developers, Chicago offers a powerful platform for professional multimedia authoring

- **Power.** Chicago's new 32-bit architecture squeezes vastly improved multimedia performance out of PCs, so developers can capture digital video and sound that is bigger and bolder than ever before. The multitasking architecture of Chicago makes it a much more convenient working environment for multimedia authors.
- **Professional Quality.** The streamlined architecture of Chicago's digital video, digital audio, MIDI and file handling subsystems enable authors and toolmakers to create very high-quality, sound, video, and animation effects. Chicago is a very attractive platform for professional development of multimedia effects and footage beyond the realm of the PC − TV commercials, for example.

For hardware makers, Chicago offers exciting new opportunities

- **Graphics.** A display driver technology called Display Control Interface (DCI) offers ways for Windows to take advantage of hardware assistance for several graphical operations such as image stretching.
- **Sound.** A new technology called Polymessage MIDI offers sound card makers a way to play very, very complex MIDI sequences with virtually no CPU use. Sound cards are improving rapidly, and there is a great deal of room for competition on a feature basis.
- **Other features.** Joystick support in Chicago raises the importance of a digital joystick port in consumer PCs. An improvement in modem technology called VoiceView offers opportunities for making computer telephony easier and more appealing for customers.

Making Multimedia Easier

Plug and Play Support

As multimedia applications, titles, tools, and games have become more and more compelling, consumers have begun buying add-on multimedia components (such as CD-ROM drives and sound cards). Buying these devices has been cheap and easy;

installing them has been a different matter. To put it mildly, installing a CD-ROM in a PC has heretofore required... patience.

Chicago's new Plug-and-Play standard will make the prospect of adding a new multimedia device to a PC considerably less daunting. Just plug in a Plug and Play enabled sound card and (literally) it plays. In fact, Chicago even makes the prospect of installing *old* multimedia devices less daunting − Windows Chicago includes tools that make it vastly easier to identify and resolve conflicts between so-called "legacy" devices that are not plug-and-play enabled. Chicago includes built-in drivers for the most popular sound cards to make this process as painless as we can possibly make it.

It is difficult to overstate the importance of plug and play for multimedia. Plug and play will do three things for the multimedia market:

- It will allow the base of multimedia capable PCs to grow through plug-and-play upgrade kits, rather than placing so much of the growth burden on the purchase of new CPUs. Because Chicago includes the basic architecture for handling sound, MIDI, and digital video, every Chicago PC can easily be made into a multimedia PC − just plug in a sound card and/or CD-ROM drive.
- It will substantially diminish the cost of installing and supporting multimedia devices, which will (among other things) help speed their adoption for business use.
- As multimedia standards (such as CD-ROM speed) continue to improve, plug and play will allow consumers to upgrade multimedia components conveniently without replacing their entire PC. Plug and play support will be vital for adoption of new multimedia devices like MPEG cards.

AutoPlay: Spin and Grin

In various ways, titles and games that run off a CD-ROM feel a bit different than other applications. For one, the way to start a CD-ROM program differs from hard disk-based applications—you first have to open your drawer, extract the right disk, and place it in the CD-ROM drive. Then you can run it like any other program. Assuming, of course, that you can find the icon you created when you first installed the program—a second difference between CD-ROM products and hard disk-based applications is that CD-ROM products may be used less regularly.

In watching users run multimedia applications, we realized that the act of placing a disk in a CD-ROM drive is loaded with information. If the CD-ROM is a title, and you've never run it before, then the act of putting it in the drive means that you intend to install the program. If you already have installed the title, then the act of inserting the disk means that you intend to run it.

Simple enough. In Chicago, we have implemented a feature called AutoPlay that allows software developers to make their products easier for customers to install and run. When you put a disk into a CD-ROM drive, Chicago automatically spins it and looks for a file called AUTORUN.INF. If this file exists, then Chicago opens it and follows the instructions.

This new feature will make the setup instructions for a Chicago-based multimedia game or title almost absurdly easy:

> To play ______, insert the disk in your CD-ROM drive.
>
> Have a nice day!

Built-in Support for Digital Video

For the past several years, Microsoft has been developing a high-performance architecture for digital video − Microsoft Video for Windows. (For more details, see the "Multimedia Graphics Architecture" section later in this chapter.)

In the past, Video for Windows was distributed separately (principally as a Software Developers' Kit). With the release of Chicago, Microsoft Video for Windows is built right into the operating system. For the first time, the ability to play digital video will be built into every copy of Microsoft Windows (including Windows NT). This has several implications:

- Users and independent software vendors can use the .AVI file format to distribute digital video files with the same confidence that they today distribute files of other Windows-supported formats like .TXT, .WRI, .BMP, .PCX, and .WAV.
- The barriers to entry for would-be multimedia title and tool developers will be further lowered because the issues of licensing and installing Microsoft Video for Windows will disappear.

Built-in Support for Sound and MIDI

MIDI is the computer equivalent of sheet music. Using sheet music, you can describe how to play Beethoven's Moonlight Sonata in a few pages − but in order to actually play the song you need to find a piano and a person who knows how to read sheet music. When you hear the music performed from the sheet music, you can expect some variation in sound depending on the circumstances − for example, if you use an expensive grand piano, the sonata will sound better than it would if you used an old upright.

Similarly, a MIDI file can contain the electronic instructions for playing Moonlight Sonata in just a few kilobytes − but in order to play the song you must have a device (such as a sound card) that knows how to "read" MIDI instructions and that can produce a piano sound. Just as the sound of pianos varies somewhat in the real world, so does the sound of a piano on a sound cards.

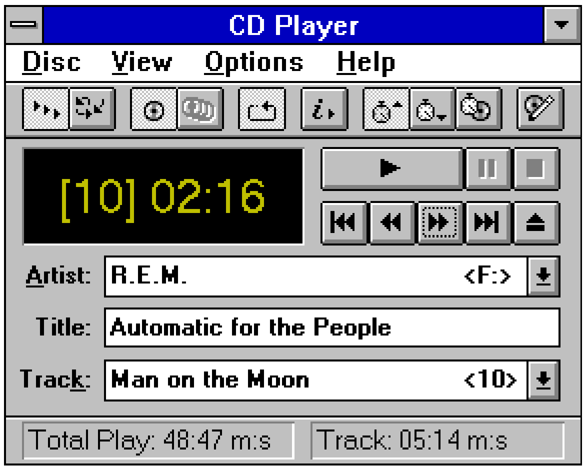
At the high end, MIDI is used as a development tool for musicians. Virtually all advanced music equipment today supports MIDI, and MIDI offers a convenient way to control the equipment very precisely.

At the low end, MIDI is becoming an ever more popular tool for multimedia product developers because it offers a way to add music to titles and games with a tiny investment of disk space and data rate. The majority of sound cards today have on-board MIDI support built in.

Windows Chicago includes built-in support for both MIDI and waveform audio (.WAV).

CD Player: Whistle while you work

Many people like to play audio CDs in their CD-ROM drives while working. So we created a CD Player to go into Chicago. The controls on this player look just like a regular CD player, and it supports many of the same features you find in advanced CD players (such as random play, programmable playback order and the ability to save programs) so that you don't have to re-create your playlist each time you pop in a CD.



[10] 02:16

Artist: R.E.M. <F:>

Title: Automatic for the People

Track: Man on the Moon <10>

Total Play: 48:47 m:s

Track: 05:14 m:s

Figure 86. You already know how to use the Chicago CD Player. Your CD will play, uninterrupted, in the background as you work.

Making Windows More Fun

Fast DIB Drawing

For the past year, the home market has been the fastest-growing segment of the PC business. More and more of our customers are telling us that they want games for Windows – and at this point, there aren't many. Games are already the largest category of multimedia application, but most of today's computer games are running on MS-DOS.

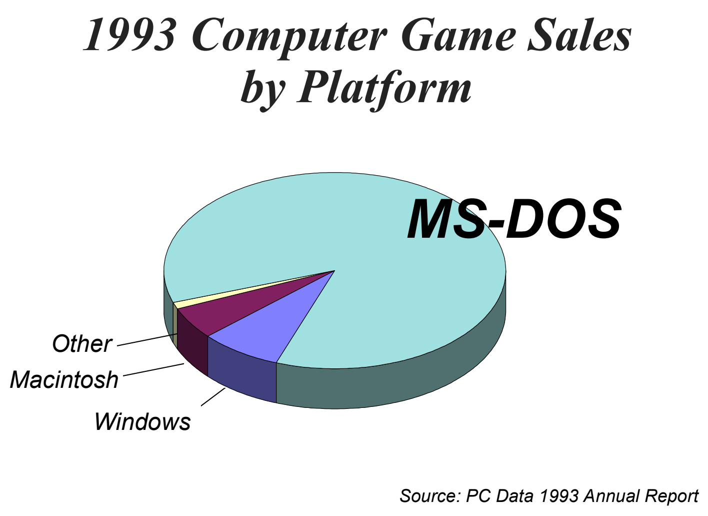


Figure 87. At the end of 1993, computer games were one of the last remaining software categories for which Windows product sales trailed MS-DOS product sales.

The speed of graphics (or, more appropriately, the lack of it) in Windows has been one of the most important obstacles keeping game developers from choosing the Windows platform for their games. We have addressed this issue head-on in Chicago in a way that provides substantially improved speed while preserving the device independence that makes Windows appealing in the first place.

A new 32-bit call was added to the Win32 API for Chicago and Daytona, called CreateDIBSection. This new feature allows developers to get bitmaps onto the screen as quickly as possible − if there is nothing fancy (such as clipping or stretching), the CreateDIBSection call will actually allow applications to send DIBs more or less directly to the video frame buffer. (For more information, see the diagram in the "Multimedia Graphics Architecture" section later in this chapter.)

Because we recognize that this kind of graphic speed is critically important to quality games, we have moved a portion of the Chicago CreateDIBSection improvements into a tool for Windows 3.1, called the WinG libraries. WinG (pronounced "Win Gee" − the "G" stands for games) libraries allow game developers to create fast, graphical games for Windows today with the assurance that your game will be fast and compatible with Chicago.

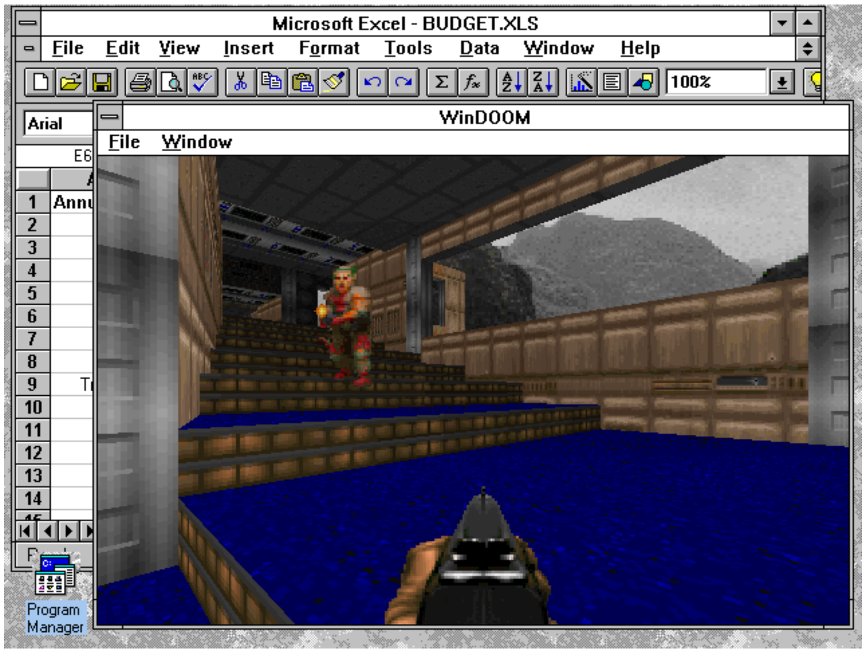

Microsoft Excel - BUDGET.XLS
File Edit View Insert Format Tools Data Window Help
100%
Arial
WinDOOM
File Window
Program Manager

Figure 88. The graphics core of DOOM for Windows was ported from MS-DOS to the WinG library in 2 days. Id Software will ship a full version of the product sometime this year.

Built-in Joystick Support

Not much to say here, really. Chicago has built-in joystick support in Chicago. *Have fun!*

Making Multimedia More Engaging

Bigger, Faster, Better-looking 32-bit Digital Video Playback

Displaying digital video involves moving and processing huge streams of data continuously and efficiently. Chicago's new digital video implementation offers some exciting new efficiencies that will allow software developers confidently to create multimedia titles that are more compelling and good-looking than ever before.

Multimedia title and game developers are business people − when they create a product, they do so with the hope of turning a profit. To maximize the number of PCs that can run a title, most developers tend to include lowest-common-denominator digital video. To ensure that as many PCs as possible can play their title or game, developers have tended to use "postage-stamp" sized video windows with low frame rates (which make movement look "jerky") and extreme compression (which makes the video look "blocky").

Chicago will raise the lowest common denominator significantly.

In the past, the process of displaying digital video has relied on a series of 16-bit systems − from reading data from the disk, to decompressing the video data, to displaying it on screen. One key design goal of Chicago was to transition this architecture to 32 bits, and the difference is eye-popping. For multimedia users, installing Chicago will be the quickest and cheapest multimedia upgrade available. Without adding any hardware, Chicago enables customers to display bigger, smoother, more colorful digital video than ever before.

It's also important to note that Chicago multimedia is fully compatible with 16-bit multimedia titles. Early testing has shown that the 32-bit improvements in file access speed and stream handling results in performance improvements even for 16-bit multimedia applications − the biggest improvements, of course, will be realized in the new generation of fully 32-bit titles that will be designed for Chicago.

For customers who upgrade their PC to Chicago, one easy-to-overlook source of performance improvements is the display driver. Many display drivers are updated more or less continuously, whether to fix problems, enhance performance, or to incorporate new features such as DCI. Most customers, however, don't update drivers on their system unless they are having a problem. Upgrading to Chicago will ensure that they have the latest and greatest.

Multitasking and Threads: "We *Don't* Interrupt This Program..."

Multimedia applications don't take well to interruption. − When you are watching a video clip or listening to a sound file, you really don't want it to stop in the middle.

The multitasking in Chicago is quite different from prior versions of Windows because it is preemptive. In Chicago, multiple 32-bit processes can share the CPU at the same time, whether those processes have been initiated by different applications ("multitasking") or by one application ("threading").

This has a very important implication for how multimedia titles will feel to consumers. Threading allows multimedia titles and games to have a more smooth, finished feeling to them. For example, a game might have a thread that plays background music continuously during game play. This would help smooth out the breaks between scenes, when the game is loading new data on another thread of the program.

There is at least one other benign externality for multimedia in the move to 32-bitness. As applications, tools and codecs are gradually rewritten to 32 bits, video and other multimedia processes will become less and less likely to be interrupted by other applications. A simple example of this is that in Chicago you can move a video window while it is playing without interrupting it.

Built-in Support for Fast CD-ROMs

The trend toward faster CD-ROM drives (double- and triple-speed) is a very good thing for multimedia computing. To get the best possible performance from these new devices, Chicago includes a new 32-bit CD-ROM file system (CDFS) for reading files from CD-ROM drives as quickly and efficiently as possible. The Windows 3.1 system for reading files from CD-ROM drives (MSCDEX.DLL) will be included in Chicago for last-resort compatibility with products that rely on it.

Faster reading of CD-ROM data helps to make video and audio playback from CD-ROM drives look and sound better. This is an important component of the overall performance enhancements to multimedia in Chicago.

Chicago also extends its support for CD-ROM to drives that read XA-encoded disks, such as Kodak PhotoCD and Video CDs.

MPEG−Hardware Support for TV-like Video from Your CD-ROM

MPEG is a very complex codec (compression/decompression system) for squeezing digital video and stereo audio into an incredibly small data stream. For example, most feature movies can fit on two CD-ROMs with MPEG compression.

Because MPEG is so complex, displaying video from an MPEG file is a very calculation-intensive process − so calculation-intensive, in fact, that the only practical way to display MPEG video on today’s PCs is by using hardware assistance.

Together with the Open PC MPEG Consortium, Microsoft has defined an industry standard for MPEG board and chip makers that want to ship MPEG devices for Chicago. This standard will allow applications to incorporate MPEG video without worrying about precisely which vendor’s MPEG device is present to decompress it.

Powerful Development Environment

Sound Compression for CD-quality Sound

Sound can take up a lot of disk space. Full CD-quality, uncompressed audio contains a lot of data—about 176K for every second of sound! An entire CD-ROM can contain only a little over an hour of music. It can also eat up a fair-sized chunk of the data rate that a CD-ROM drive is capable of sustaining.

To lessen the burden of storing and playing sound from an application , Chicago includes a family of sound compression technologies (“codecs”). These codecs can be divided into two groups:

- Music-oriented codecs (such as IMADPCM) are included that allow close to CD-quality sound to be compressed to about one-quarter size.

- Voice-oriented codecs (such as TrueSpeech™) are included to allow very, very efficient compression of voice data.

This support for compressed sound is two-way − you can play sound from a compressed sound file, or you can compress a sound file (using the built-in sound recording and editing utility). If you have a microphone, you can turn on voice compression when recording so that your file is compressed in real time.

In addition to the codecs that come with Chicago, the audio architecture of Windows multimedia is designed to be extendible through other installable codecs. The video architecture of Windows multimedia can be extended in the same way.

Polymessage MIDI Support for Better Sound

Chicago comes with Microsoft's best-ever implementation of MIDI, including a new technology called polymessage MIDI support. This enhancement allows Microsoft Windows to communicate multiple MIDI instructions simultaneously within a single interrupt. The result of this change is that playing MIDI files now requires even less computing power than it did before, and allows developers to process MIDI instructions alongside graphics and other data even more successfully.

Multitasking

Multitasking makes Chicago a much more attractive platform for multimedia authoring. Creating multimedia content is very CPU-intensive work that can take a long time to complete. For example, compressing a digital video file can take hours, depending on the complexity of the file and the specs of the system doing the compression. Currently, authors who want to compress more than one digital video file have to do them one at a time − when one file finishes, they can start the next. The result was that video authors were virtually chained to their desks until late at night.

With Chicago, authors still have control of their PC, even when an enormous compression operation is underway. This makes it possible for digital video authors to initiate several compression operations at once − and then head home.

Professional Quality

Capture and Compression of Bigger Digital Video

When it comes to capturing digital video, there's no avoiding the grim reality that video contains an enormous amount of data. Capturing digital video is even more data-intensive than playing it back, because raw digital video footage is uncompressed. A single frame of full-color video at 640x480 contains close to a megabyte of data. At 30 frames per second, you can fill up a 1 gigabyte hard drive with uncompressed video data in less than a minute. There are ways to compress this data in order to make your storage go further, but no matter what, the rate at which you can write data to disk is of great importance.

The 32-bit file access of Chicago is therefore every bit as important to digital video authors as it is to digital video users. Because you can write more data to disk more quickly in Chicago, you can capture better-looking video—bigger, more frames per second, more colorful.

Once the raw footage is captured, the next step is to compress it − a potentially time-consuming process. Both Cinepak and Indeo™ will be available in 32-bit versions for Chicago, and this should make the process considerably more efficient.

General MIDI: You Want a Trumpet, You Get a Trumpet!

One of the early challenges for MIDI was that it was, in a way, too flexible. Any instrument can be "connected" to any MIDI channel, so that a "sequence" (song) written for a piano might accidentally end up being played on a tuba.

The general MIDI specification is an industry standard way for MIDI authors to request particular instruments and sounds. Microsoft supports this standard.

Built-in Support for Multimedia Devices

Chicago includes built-in support for common multimedia authoring devices like laser disks and VCRs. This makes it easy to set up a system for "step capture," a process in which the author captures digital video data one frame at a time, usually to be compressed later. This is a slow process, but it is absolutely the best way to get the best possible quality digital video.

Frame-accurate control of the VCR is also important for recording broadcast-quality special effects to use in commercials, movies, television programs, music Multimedia PCs for 1995 videos, and the like.

Opportunities for IHVs and OEMs: Multimedia PCs for 1995

All things being equal, installing Chicago will upgrade any PC into a more capable multimedia tool.

Of course, all things are *not* equal. There is a great deal of variation in the quality and capability of multimedia PCs and devices. The *Microsoft '95 Hardware Design Guide* is being published under separate cover to help IHVs and OEMs identify opportunities to take advantage of new capabilities in Chicago.

We are making five high-level recommendations to OEMs:

- **Balance beats horsepower**. Multimedia playback places heavy demands on many parts of the system, from the CD-ROM (reading data) to the hard disk (writing data) to the CPU (decompressing) to the video and audio subsystems (playing it). A fast CPU does not guarantee a great playback system. In fact, multimedia playback on most of today's high-end PCs is not constrained by the CPU.

- **Local bus video is indispensable**. Even OEMs creating "non-multimedia" systems should use local bus video, because doing so will enable consumers to plug-and-play their way to a multimedia system later, should they choose to do so. Without local bus video, a PC will not be able to keep up with the amount of video data that 1995's consumer multimedia titles and games will want to display continuously.
- **Double-speed CD-ROM or better**. Titles in 1995 will be written assuming double-speed data rates.
- **SVGA (800x600) or better with 16-bit color**. Why more colors than 256? Because multimedia applications use a lot of colors, and tend to compete for access to the system palette. Consider the challenge of a multimedia presentation that includes a digital video clip of an underwater scene on a slide with a smooth-shaded maroon background. There aren't enough colors in a 256-color palette to make both the slide background and the underwater scene look good.
- **16-bit audio**. The installed base of sound cards that can interpret MIDI is now large enough to be tempting to game and title developers. Not all sound systems are equal − some sound great (16-bit with sampled sounds), and some sound like Star Trek reruns. The differences are significant, and customers will be able to tell the difference.

New Opportunities for Great-Sounding Audio

There is a great deal of variation in the quality of audio cards and sound systems. Most of the time, sound cards up to now have been used principally for their ability to play waveform audio − the equivalent of recorded sound. For some uses, like voice-overs, there is no realistic alternative to recorded waveforms. However, recorded sound is very resource-intensive for both the CD-ROM and the CPU. In Chicago, there are enhancements to the handling of MIDI that makes it an even more appealing alternative to .WAV for playing music within games and multimedia titles. There are several things that makers of audio cards and systems can do to distinguish themselves:

- **Polymessage MIDI support.** This is a very efficient new technology included in Chicago that makes it easier for application and game writers to use MIDI. If a sound card supports polymessage MIDI, the CPU use required to play even a very complex song is quite small.
- **16-voice or better polyphony.** Polyphony is the ability to play multiple sounds at once. Support for more concurrent sounds means fuller-sounding playback.

- **Sampled sound rather than wave-form synthesis.** Wave-form synthesis uses a mathematical approximation of a sound, such as a piano. Sampled sound is an actual recording of the piano, and sounds considerably better. Including samples of at least the most common General MIDI instruments helps ensure that music in games and titles sounds really good, instead of synthetic.

DCI: Taking Advantage of New Video Card Features

In the summer of 1994, Microsoft will release the new DCI display driver. This technology was developed in partnership with Intel and other makers of advanced video display cards.

DCI is a device driver level interface that allows Windows to take advantage of hardware features that are (or could be) built into advanced display adapters, specifically:

- Stretching which speeds up rendering of images that are stretched or distorted.
- Color-space conversion which assists in playback of compressed digital video by accepting YUV data instead of requiring RGB.
- Double Buffering which allows faster, smoother block transfers (BLTs) of images by providing memory space for off-screen drawing.
- Chroma key which facilitates the merging of video data streams, allowing a particular color to be treated as “transparent” in the merge operation.
- Overlay which speeds display of partly concealed objects.
- Asynchronous drawing which, along with double buffering, provides a faster method for “drawing” into off-screen memory space.

Most of the hardware features above relate to the fast, efficient decompression and playback of digital video. Applications that use the Microsoft Video for Windows architecture will benefit from these features automatically and substantially.

Multimedia Architecture

Multimedia Graphics Architecture

There are four kinds of graphics an application might want to “draw” on the screen, and four APIs that an application can use to do so:

- **“Productivity application” graphics**. Scroll bars, fonts, buttons, and the like. Applications that want the system to help them draw these things use GDI, the basics Windows graphics API.
- **Digital video**. Applications that want to play digital video use the Video for Windows API. More details on the Video for Windows architecture are provided in the following section.

- **Game graphics**. Games draw their own graphics (in memory) and want bitmaps blasted to the screen as fast as possible. That's what WinG does. It is available for Windows 3.1, and provides many of the same benefits of Chicago's CreateDIBSection function, as well as fast access to the frame buffer through DCI.

- **3D engineering graphics**. Applications that want the system to help them draw 3D solids use OpenGL. OpenGL is Microsoft's strategic choice of 3D application programming interface. We have a long- term commitment to deliver an implementation of OpenGL as part of the broader Win32 API, a commitment we announced last November. Our first OpenGL implementation will ship in Windows NT "Daytona."

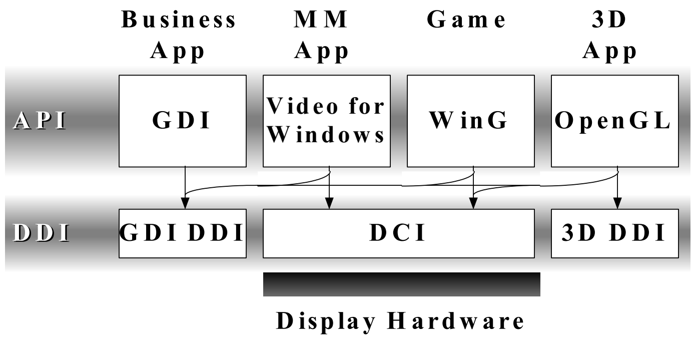


Figure 89. Windows Graphics Architecture

There are three pieces to the device driver interface in Windows, and the APIs described above are designed to take advantage of whatever DDI provides the best performance.

- GDI-DDI is the basic graphics device driver interface for Windows. It is optimized for the flexible graphics requirements described above for the GDI API.

- DCI is the new device driver interface created jointly by Microsoft and Intel. DCI drivers provide a fast, direct way for games and digital video in windows to write to the video frame buffer. It also enables digital video playback to take advantage of several specific kinds of hardware support included on advanced graphics adapters. For example, stretching hardware can allow users to scale up the size of a digital video clip with virtually no additional strain on the CPU. Color space conversion support in hardware can reduce the amount of work a codec must perform by up to 30%, allowing substantially better video playback.

- The 3D-DDI is still in the process of being defined, and should be completed in the summer of 1994. This DDI will enable applications that use OpenGL to take advantage of accelerated 3D support in hardware.

How Multimedia Data is Routed in Windows

The following diagram describes (in simplified form) the path that synchronized multimedia data travels from storage to experience during playback.

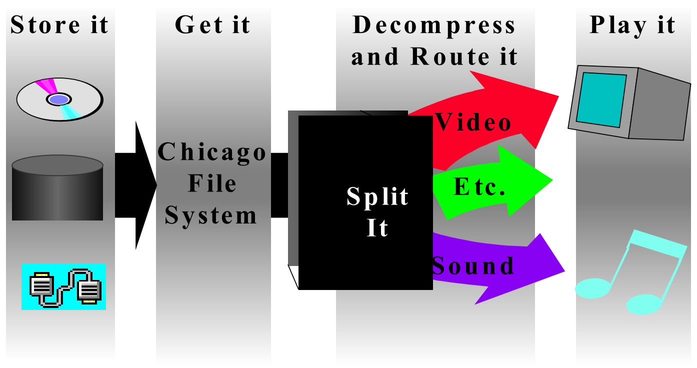


Figure 90. Windows Multimedia

First, the data (usually an .AVI file) must be stored somewhere, such as a CD-ROM, a local hard drive, a network file server, or another storage medium. The quality of the eventual playback will be constrained by the amount of data that the storage medium can supply to the file system continuously.

A command (such as Play), usually issued through the Media Control Interface (MCI), causes the relevant part of Chicago’s file system to retrieve the stored data. Obtaining this data swiftly and steadily is vital to the success of overall playback performance, and the 32-bit protected-mode enhancements in Chicago’s new file system (and CDFS) have a lot to do with the overall performance enhancements in Chicago multimedia.

A multimedia data stream (such as an .AVI file) generally contains multiple components, such as digital video data, audio data, text, and perhaps other data (such as hot spot information, additional audio tracks, and so forth.) As multimedia information comes off the CD-ROM, the first job of the Video for Windows architecture is to figure out what the data stream contains, and to separate and route it accordingly.

In most cases, digital video and digital audio are stored in a compressed form. Before it can be seen or heard, therefore, it must be decompressed. Frequently, this function is performed in software. If hardware support is available on the graphics adapter or sound card for all or part of the decompression work, however, Video for Windows can tap into it.

The Video for Windows architecture has been created in a way that allows *installable* codecs. Chicago ships with a set of useful software-only codecs for both video and audio, but you are not limited to these tools only. As new codecs become available for particular audio and digital video needs, they can be plugged into the Video for Windows architecture. For example, motion JPEG is a useful codec for multimedia authoring − capture cards that support JPEG compression and decompression are easily available, even though JPEG itself is not explicitly provided in Chicago.

Installation and Setup of Chicago

The very first contact that users will have with Chicago will be during the time they initially install it on their computer. If the Setup process is not easy, or the user is confronted with a series of configuration-related questions that they don't understand how to answer, the initial experience with an operating system for a novice or intermediate user will be bad, and will set the tone for their initial trial. Advanced users can overcome difficult installation procedures, but their frustration level will still have a finite threshold.

Setup in Chicago is completely rewritten to offer greater flexibility and better customization than Windows 3.1 does. In addition, Setup in Chicago is more modularized than Setup in Windows 3.1, allowing the easy customization of individual Setup steps, as well as the easy installation of new custom components.

Summary of Improvements over Windows 3.1

Setup of Chicago has been improved over Windows 3.1 in a number of areas, including:

- A modular setup architecture that provides increased customization and flexibility
- An entirely GUI-based approach and improved interaction with the user, including better visual feedback of progress during setup
- Improved hardware device detection and configuration support
- Better customization over components to install
- Built-in smart recovery mechanisms for failed setup
- Built-in verification of installed components for easy correction and replacement of corrupted or deleted files
- A network setup process which is well-integrated with other setup components, and provides support for a number of network installation configuration scenarios
- Support for an automated batch installation procedure, allowing Chicago to be installed with little or no user intervention
- Better flexibility for PC installers, VARs, and MIS organizations to customize Setup by adding components to be installed at setup time, such as custom in-house applications or other solution offerings

Modular Setup Architecture

Setup in MS-DOS is responsible for installing the basic disk operating system on the PC. Setup in Windows 3.1 is a combination of components and installation procedures inherited from prior versions, and is responsible for installing the GUI on the PC. Setup in Windows for Workgroups extended the Setup functionality in Windows 3.1 to install networking components on top of the GUI and disk operating system. As Chicago is a complete, integrated operating system, it is now responsible for installing the disk operating system, the GUI, and the networking functionality on the PC. This posed some interesting problems when the Chicago development team first approached the daunting task of writing Setup for Chicago.

The original Setup written for Windows was not flexible enough easily add additional components to the setup process, without making the installation procedures unwieldy. To make the installation process easier, modularized, and more flexible, the Chicago development team for Setup completely rewrote the installation code. Chicago also incorporates the use of more intelligent defaults and mechanisms for automatically configuring or installing key components while requiring only minimal user intervention, furthering the ease-of-use of the operating system.

For end-users, Setup in Chicago provides a simple, easy way to initially install and configure Chicago. For MIS organizations, Setup in Chicago provides greater control and flexibility over components that are installed, and offers support for automated batch installs to further simplify the setup procedure.

GUI-based Setup Program

Setup in Chicago differs from that of Windows 3.1 by featuring an entirely GUI-based setup process. Using a GUI-based setup simplifies the interaction with the user by providing better visual feedback of configured options, and greater flexibility for navigating through the setup process. To support a GUI-based setup, Chicago features a Setup program that runs entirely from within the Windows environment. Users who have either Windows or Windows for Workgroups on their PCs already, Chicago Setup run like a Setup program for a Windows-based application. For new installations, Chicago Setup includes the necessary components to install a minimal version of Windows to support the GUI-based setup process.

The GUI-based Setup also provides better visual feedback to the user throughout the installation process. Users are constantly shown where they are in the setup process and are given a number of visual cues that the system is proceeding with the setup process.

Leveraging of Detection Code

The modular architecture of Setup also allows the leveraging of detection and installation procedures beyond the initial setup process. The same procedures and detection mechanisms used by Setup to detect and initially configure hardware devices and peripherals in the Setup process, are also used for maintaining or detecting the devices post-Setup. For example, the same code base used during Setup that for the detection of Plug and Play or legacy hardware devices is also used to detect or configure new devices once Chicago is up and running.

Improved Customization

Windows 3.1 provided few mechanisms for easily customizing the setup process, but Chicago makes customization easier for system administrators. Customization of Setup allows for better control over components installed into an existing environment. MIS organizations can now easily tailor the existing configuration options for Setup components such as supported network interface cards, or supported printers. Chicago also offers the flexibility for system administrators to add on components to be installed during the setup process or to run additional procedures during the final phases of Setup.

Improved Hardware Detection

During Setup, Chicago detects the different hardware devices and components configured on the computer and uses this information to install drivers and set the appropriate entries in the Registry.

Unlike the simple hardware detection mechanisms used in Windows 3.1 to identify the PC configuration for a narrow group of devices, Chicago provides more versatile hardware detection and configuration mechanisms and provides detection support for a wider range of devices.

Chicago provides straight-forward detection support for the base computer components such as communication ports and processor type, but provides more robust detection of system devices including video display adapters, pointing devices, hard disk controllers, floppy disk controllers, and network interface cards.

Chicago Setup also helps to detect any hardware resource conflicts at an early point during the setup process. Hardware resources such as IRQs, I/O addresses, or DMA address in use by more than one device can cause havoc when initially installing an operating system and may prevent the system from starting properly..

Chicago detects hardware components and devices one of two ways:

- It leverages Plug and Play detection to identify Plug and Play devices and peripherals.
- It uses a manual query detection mechanism for legacy devices and peripherals.

Once setup detects the device, Chicago installs the appropriate device drivers and configures the system.

Simplified Four-Phases Setup

Chicago Setup is quite a bit simpler than Windows 3.1 Setup and is divided into four logical phases:

- Hardware detection
- Configuration questions
- Copying Chicago component files
- Final system configuration.

The following sections describe what happens in each of these phases.

Hardware Detection Phase

During the hardware detection phase, Setup analyzes installed system components, detects installed hardware devices, and detects connected peripherals. During this phase of Setup, Chicago analyzes the system to identify the hardware resources that are available (for example, IRQs, I/O addresses, and DMA addresses), identifies the configuration of installed hardware components (for example, IRQs in use), and builds the hardware tree in the Registry.

Chicago uses a number of mechanisms to detect installed hardware devices during setup. For legacy PCs, Chicago maintains a database of known hardware devices and performs a manual detection to check I/O ports and specific memory addresses to attempt to identify whether they are being used by recognized devices. Chicago will also check for Plug and Play peripherals connected to legacy PCs, which return their own device identification codes. For PCs that contain a Plug and Play BIOS, Chicago queries the PC for installed components and the configuration used by these components (Chicago also checks the system for connected Plug and Play Peripherals on Plug and Play PCs).

During the hardware detection phase of Setup, Chicago tries to identify hardware conflicts and provides a mechanism to resolve conflicts early during the installation process to overcome hardware configuration issues that Windows 3.1 users encounter.

Once the hardware detection phase is complete, the user is presented with a dialog box on the screen allowing them to proceed with Setup, or to review the hardware devices that were detected and system components that Chicago will install.

Configuration Questions Phase

Chicago uses information found in the first phase to determine which system components it will install. Chicago consolidates the configuration and customization phase of Setup into a single process at the beginning of the Setup procedure. By contrast, users were constantly asked for system configuration information and confirmations during Windows 3.1 Setup.

Users can review the components Chicago will install, and remove or add any components.

Copying Files

This phase of Setup is the most straightforward. Once the user has identified or confirmed the Chicago components to install, Setup begins copying files from the Chicago installation disks (or from a network server, if specified). Once the necessary files are copied to the user's PC, Setup prompts the user to remove any disks in floppy drives and then reboot the system to proceed with the final phase of Setup.

Final System Configuration

During the final system configuration phase, Setup upgrades the existing configuration of Windows and replaces the existing version of MS-DOS with the new Chicago operating system.

After files are updated and the system is configured, Setup guides the user through a process to configure peripheral devices such as modems or printers that may be connected to the system. Once this is done, Chicago is ready to use!

Better Control Over Installed Components

Users now have greater control over components and parts of Chicago that are installed during the Setup process. Based on the modular architecture of Chicago, users will be able to selectively choose the options that Chicago will install for the given functionality that they desire.

Smart Recovery Mechanism for Setup

With Windows 3.1 Setup, if the system hung during device detection, or if Setup procedure ended abnormally, it would set a flag disabling hardware detection for the next time that Setup ran. This mechanism provided a means for a user to by-pass a section of setup that would otherwise fail. However, to do this the user was required to rerun the entire setup procedure and manually identify hardware devices.

Chicago supports a far better recovery mechanism in the case of setup failure. During the setup process, Chicago creates and maintains a log as the setup operations are performed and the hardware devices are detected. If Setup fails, perhaps due to a hang during hardware detection, the last entry in the Setup log identifies where the process was interrupted. To recover and resume, the user simply reruns Setup—the Setup program recognizes that it was run before, and will begin from where it left off. In the case of a hang during a hardware detection procedure, the system will actually bypass the detection module where the hang occurred, and will allow the user to manually select the proper hard device installed in or connected to the system.

Built-in Verification of System Files

Under Windows 3.1, if a user accidentally deleted a component file or a system file was corrupted, there was no easy way for a user to recover the given file. A user needed either to use the Expand utility to recopy over a known file, or to completely reinstall Windows 3.1 to reinstate a lost file.

Chicago provides some very flexible solutions to this problem. During the Setup process (and subsequent maintenance of the Chicago system), Chicago creates and maintains a log of the installed components. This information is used as part of Setup's smart recovery support, and is also used to verify the integrity of installed components.

If a user runs Chicago Setup after Chicago is already installed, Setup asks the user whether to reinstall Chicago or simply to verify installed components. If the user wants to verify installed components, Setup examines the setup log and reruns through the Setup process *without* completely copying all system components. Chicago verifies the integrity of files that were installed during Setup with the files provided on the Chicago installation disks. If the integrity check fails due to either a missing file on the Chicago computer, or a file was corrupted, Setup automatically reinstalls the missing or damaged file.

This capability in Chicago greatly simplifies and reduces the time required to resolve missing files or corrupted configurations, thus helping to reduce the time and money required to support desktop configurations.

Network Setup Improvements

Chicago provides improved support for installation and use in network environments. Chicago can be installed on a network to upgrade existing Windows users, or can be used to convert existing MS-DOS PCs. Chicago offers the same capabilities for running Windows from a network, but also provides additional functionality to better address the requests of MIS organizations.

In addition to basic support for stand-alone computers, Chicago includes Setup provisions for better supporting the following installation:

- Installing and running Chicago from a local computer on a network
- Installing and running Chicago from a network server instead of installing on the local computer
- Installing Chicago on a network server and supporting diskless computers that RIPL boot from the network server

- Installing Chicago on a network server and supporting a computer with a single floppy drive to connect to the network and run Chicago from the network server

Additional information about network support in Chicago is discussed in the Networking section of this guide.

Network Installation Location Remembered

To make it easy to install new drivers when a user modifies the configuration of a PC in a networked environment, Setup in Chicago remembers the location on the network from which it was installed from. Whether the server was a NetWare server, or a Windows NT Advanced Server, when the new user adds a device or requires additional driver support files to properly run Chicago, the Setup code will automatically attempt to get the files from the network server. Setup stores a UNC pathname in the registry, eliminating the need to maintain a permanent network connection on the PC.

Any user that has been prompted for the insertion of a diskette containing needed files for Windows or Windows for Workgroups will find this new functionality a blessing.

Batch Installation Support

Chicago features a batch installation option that provides for the use of an installation script to automate the installation process. MIS organizations or VARs can simplify the installation procedure for a user by specifying answers to questions that Setup needs information for, as well as specifying defaults to use for installing and configuring devices such as printers.

The batch install capability of Chicago is more flexible and customizable than that provided with Windows 3.1 or Windows for Workgroups 3.11.

Configuration Preserved When Upgrading from Windows or Windows for Workgroups

Chicago can be easily installed as an upgrade on a PC where Windows or Windows for Workgroups already exists. During the upgrade process, Chicago uses existing configuration information to set installation defaults and will examine the contents of specific .INI files to further determine the appropriate Setup options.

Chicago preserves configuration information, such as the Program Group definitions created by the user, and will maps user interface-related features or functionality from Windows 3.1 or Windows for Workgroups to that of the new interface used by Chicago.

International Language Support

With the growth of the world-wide PC market, the use of Microsoft Windows and Windows–based applications has made PCs easier to use around the globe. Windows and Windows–based application are sold and used worldwide. This poses some unique problems for both Microsoft as an operating system vendor, and ISVs as application developers.

When a new software application or operating system intended for a world market is developed, efforts must be undertaken to *localize* the software to a given country and written language in which it will be used. In many cases, this is as simple as changing the names of menus, menu items, and strings displayed by the software to match the foreign language used in the locale. However, as the features and functionality of a software product grow, so does the complexity required to tailor the application to characteristics of the native country. Since the start of the design work for the Windows NT operating system, Microsoft has been adding to the level of support for international languages and cultural conventions in the 32-bit editions of the Windows family of operating systems.

This section discusses the localization plans for Chicago, the built-in international support for using Chicago on a world-wide basis, and the special provisions that Chicago provides for enhancing existing or developing new applications that can be used in different parts of the world.

Summary of Improvements over Windows 3.1

Support for using the Windows operating system on a global basis is improved in Chicago. Chicago offers benefits to both end-users and to software developers, which can be summarized as follows:

Benefits for Users

- **Chicago makes it easy to use multiple language fonts and character sets, and easily switch between the different keyboard layouts required to support them.**

  With the Eastern European version of Windows 3.1, a user can directly switch between only two keyboard layouts; for example, Russian and English. With the standard Latin versions of Windows, a user cannot easily switch between different keyboard layouts directly—the user has to go to Control Panel for each language switch.

With Chicago, users can easily switch between all available languages and corresponding keyboard layouts configured on their system by using the Alt+Shift key combination—making it easy for users to integrated information into a multilingual document.

- **Chicago substitutes fonts when switching between different languages if the original font is not present on the system.**

  When switching between different languages, matching fonts for the new language are substituted if the original font is not available. This allows users to be able to read and use the text for a similar character set, even if they don't have the same font that the original information was created in.

- **Proper sorting and formatting rules are used corresponding to the presently configured locale.**

  Different locales and cultures have different rules for interpreting information. For example, cultures use different sequence algorithms for sorting information, use different comparison algorithms for finding or searching for information, and use different formats for specifying time and date information. Win32–based applications that use the National Language Support (NLS) APIs allow users to easily exchange information on a global basis, while preserving the integrity of the information.

Benefits for Developers

- **It's easy for application developers to add international language support to their applications.**

  Developers can now use the Win32 NLS APIs for sorting, searching, and manipulating information in a locale-independent way. NLS services in Chicago ensures that information is handled properly for the given culture or locale. The correct national format is automatically supplied based on the international settings specified by the user in Control Panel. For example, to obtain the current date format information to match the current locale, a developer can simply make the application call an NLS API and the system will return the proper format. Likewise, to sort information in the proper sequence in French, Norwegian, or Spanish, the application calls a corresponding culture-independent NLS API.

- **Chicago provides services for application vendors to automatically switch between the proper fonts and keyboard layouts as users navigate through a multilingual document**.

  For users who create or edit multilingual content in their documents (for example, translators), a Win32–based application that uses the international services in Chicago automatically activates the correct fonts and corresponding keyboard layouts for the edit point in the text. This allows easy editing of information contained within multilingual documents.

- **Information can be passed through the Clipboard to other applications and preserve language-specific attributes.**

Chicago provides additional services for application vendors to easily exchange information between internationally-aware applications, while preserving all language formatting characteristics.

- **Multilingual-aware applications can change the given language of the system under program control.**

  Chicago provides services that application vendors can use to automatically switch the language that the system uses to match attributes in a document. For example, as a user scrolls through a multilingual document, the application can automatically switch the system language to match the format of the information contained within the document.

- **Language-relevant information can be saved in Rich Text Format (RTF) from a multilingual-aware application.**

  Extensions are provided to the RTF specification to support storing international language information in RTF files.

Localization of the Chicago Operating System

As a result of success of Microsoft Windows around the world, Windows and Windows–based applications have been localized into many different languages. Microsoft Windows 3.1 was localized into more 25 major languages. However, this process took as long as 18 months, thus delaying the availability of Windows 3.1 for some language versions. The Chicago development team has been working on international localization issues concurrently with the development of the domestic U.S. version of the operating system. To better support a global market, Microsoft plans to localize Chicago into at least 28 different language versions. The localized versions of Chicago will be released on a planned development schedule that does not exceed 180 days. The planned localized product versions for Chicago include German, French, Spanish, Swedish, Dutch, Italian, Norwegian, Danish, Finnish, Portuguese, Japanese, Chinese, Korean, Russian, Czech, Polish, Hungarian, Turkish, Greek, Arabic, Basque, Hebrew, Thai, Indonesian, and Catalan (as well as several other variations of these languages).

International Language Issues

For an operating system to be used effectively in a world-wide environment, the localization is only a small part of the solution to offer global customers a solution. A world-wide operating system release must also provide services to support the use of international applications and support a global market by making the application developer's job easier.

Here are some international language issues which international end-users and applications developer face:

From the End-User's Perspective

Some users need to use more than one language in a document. For example, they might be translating from English to Russian or they might be writing an instruction document for a product in many different languages. This causes series of obstacles for the user. For example, the user must repeatedly switch to another keyboard layout on-the-fly so that he or she can continue writing text in a different language. Likewise, when using a database, the user faces the problem of sorting the information in the proper order for a given language.

From the Developers Perspective

When localizing a product into different languages, a developer is faced with several questions, such as: What is the correct sorting order for French? How is a date represented in Germany? Do the Swedes really need to have the ability to use the characters: Å, Ä, Ö? If a document contains text in more than one language, is there some way for the software to "know" which part of the document is in which language? Can information in a multilingual document be passed to another application via the Clipboard?

It is of course not expected that every developer knows how to address these issues, but many try natively as part of their application and come a little short, creating problems for the end-user, their support organization, and their own development team.

In Chicago, Microsoft has set out to offer international language support at the operating system and API level to add functionality which provides solutions to using software and exchanging documents around the world.

International Language Support

Since the mainstream Windows operating system platform has not previously offered international language support as an operating system service, many application vendors have hard-coded global characteristics into their applications. This allows their applications to be used in a given locale, but prevents them users from easily using the application in a different cultural environment. Therefore, the user is dependent upon the application vendor for providing a version of the application that matches their locale attributes.

Providing international language support services in the Chicago operating system makes it easier for application developers to solve international language issues related to presenting or manipulating information in their applications.

Date and Time Formats

Date and time information needs to be represented in different formats depending on the locale where the information is being used. For example, date information presented in English places the day between the month and year as in “March 9, 1994,” whereas a different locale may represent the same date as “9 March 1994.”

Sorting and Searching Support

International language issues are much more complex than simply representing date and time information in the proper format. Sorting and searching algorithms in applications must correspond to the proper language rules for the locale in which the information is being used and manipulated.

Here are some examples of subtle differences between different language rules:

- In French, diacritics are sorted right to left instead of left to right as in English.
- In Norwegian, some extended characters follow the Z character as they are considered unique characters rather than characters with a diacritic.
- In Spanish, CH is a unique character between C and D, and Ñ is a unique character between N and O.

In Windows 3.1, many developers developed their own sorting routines for different languages and hard-coded this functionality into their code. This makes their applications inflexible to support the numerous right sorting tables required for all the languages in which they want to localize their application in.

As a further example, when sorting a database in Swedish with an English-language sort algorithm, names would be sorted like this:

English sorting	Correct Swedish sorting
Andersson	Andersson
Åkesson	Karlsson
Ärlingmark	Magnusson
Karlsson	Turesson
Magnusson	Åkesson
Turesson	Ärlingmark

The system treats the Å and Ä as an A and therefore sorts it as an A at the top of the list. But in the correct Swedish sort order, the Å and Ä are sorted at the end of the alphabet, after Z. The reason that the names starting with Å and Ö are sorted as last in the correct Swedish sorting is that the Å and Ä are separate vowels in the Swedish language and occur at the very end of the alphabet. A Swede looking for “Ärlingmark” would be quite confused to find it near the beginning, not the end, of a list of names, for example.

Support for Different National Character Set Support, Keyboards, and Fonts

In standard Windows 3.1 there is no way to use fonts native to the Eastern European countries such as Greece, Russia, or Turkey. For instance, if a user tries to install a Russian font on an English or French version of Windows 3.1, the characters appear unintelligible on the screen and the user is unable to use the font. Therefore, a special English Eastern European version of Windows 3.1 was designed for English users who needed to use the Eastern European fonts, including Russian Cyrillic or Greek. The English Eastern European version of Windows 3.1 offered the same capabilities as the true Russian or Turkish EE (Eastern European) version of Windows for displaying font and character information.

International Language Solution: Multilingual Content Support

Chicago resolves many problems related to international language issues by integrating multilingual content support in the core of the operating system. Chicago also offers national language support to software developers as a series of APIs that are part of the Win32 API set.

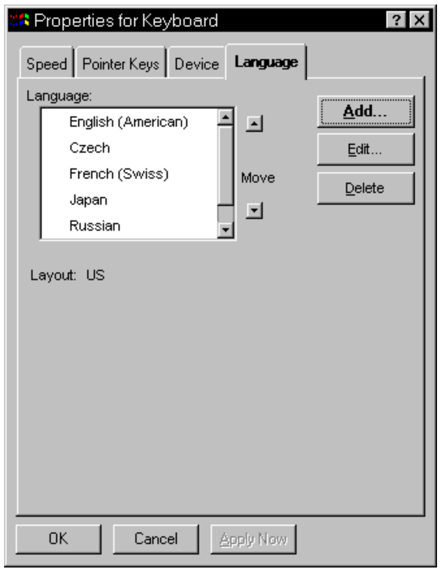
What is Multilingual Content?

Multilingual content support is the ability to display and edit text of various languages and scripts in a single document. Multilingual content support is a core feature in the Chicago product and will be supported in the next major release of Windows NT (code named “Cairo”).

Multilingual content support in an application has two major benefits. The first is that users who need to deal with content in multiple languages and scripts and exchange these documents with users on other language systems can do so. This is an important feature within the European Union, for example, where Greek- and Latin-based languages must coexist in documents. The second benefit is that an application which supports multilingual content will support the native content of any market into which it is sold. A multilingual application is a great application for the world.

Switching Between Languages and Keyboards the Easy Way

Chicago allows the user to add support for multiple keyboard layouts to match different international conventions. In Control Panel, the Keyboard icon provides the ability to configure the system to support the preferred keyboard layouts as shown in Figure 92.



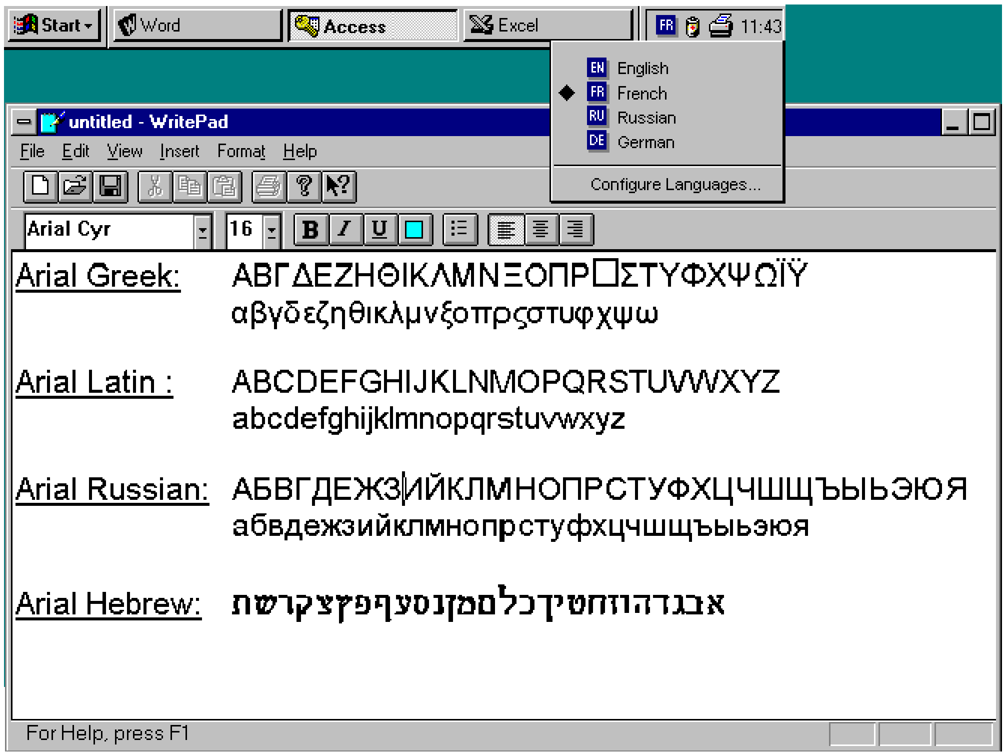
Language:

English (American)
Czech
French (Swiss)
Japan
Russian

Layout: US

Figure 91. Keyboard Properties Dialog Showing International Layout Support

Under Windows 3.1, to change the keyboard layout a user would go to Control Panel *each time* he or she wanted to change to a different keyboard format. Chicago makes this even easier. Figure 93 shows a sample legacy word processing document that illustrates the ability to integrate text using the Arial font in different languages within the same document. The language identifier in the status area of the Taskbar allows the user to easily switch the system language between the available language options. A Chicago application that uses NLS APIs would incorporate the ability to switch the preferred language directly on the toolbar of the application.



Arial Greek: ΑΒΓΔΕΖΗΘΙΚΛΜΝΞΟΠΡ□ΣΤΥΦΧΨΩΪΫ
αβγδεζηθικλμνξοπρςστυφχψω

Arial Latin : ABCDEFGHIJKLNMOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz

Arial Russian: АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ
абвдежзийклмнопрстуфхцчшщъыьэюя

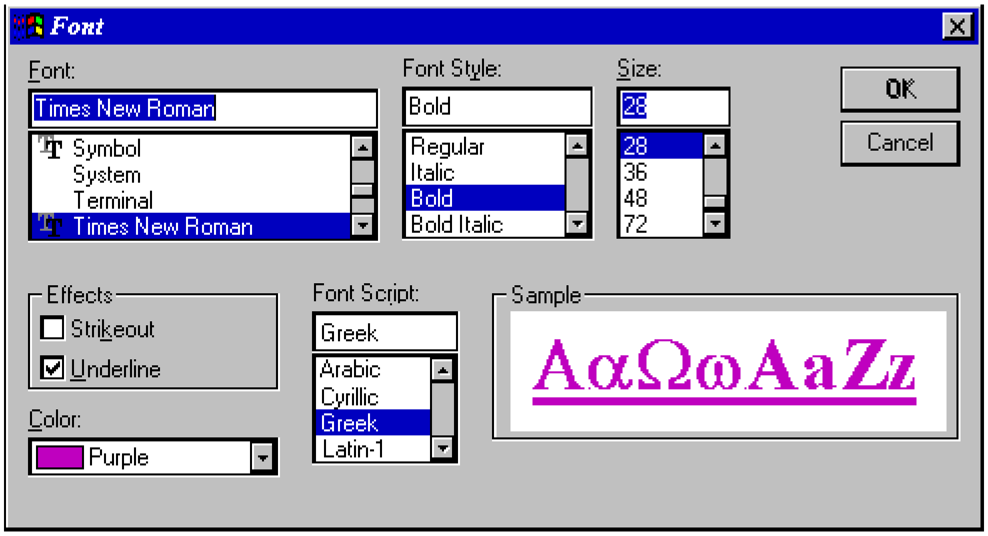
Arial Hebrew: אבגדהוזחטיךכלםמןנסעףפץצקרשת

Figure 92. Chicago Makes it Easy to Switch Between Different Languages to Create Multilingual Documents

Multilingual Extensions to the ChooseFont Dialog Box

The ChooseFont common dialog is now enhanced to include a list box showing the character set scripts supported by a particular font. This allows for proper representation of fonts for a given language.

Figure 94 shows an early representation of the new ChooseFont common dialog box, illustrating the integration of font script selection options. The scripts list shows the script names for each of the character sets covered by the font selected in the Font control. The preview window displays a font sample that is dependent on the script selected, as well as the other font attributes. The sample preview string is specific to the character set chosen by the user, showing what each of the different scripts look like.



Font: Times New Roman (Symbol, System, Terminal, Times New Roman)

Font Style: Bold (Regular, Italic, Bold, Bold Italic)

Size: 28 (28, 36, 48, 72)

Effects: ☐ Strikeout ☑ Underline

Color: Purple

Font Script: Greek (Arabic, Cyrillic, Greek, Latin-1)

Sample: AαΩωAaZz

Figure 93. Chicago Displays the Available Font Selections for a Given Font Script Chosen by the User

Internationally-aware applications can support multilingual font selection by using the ChooseFont common dialog box (thus allowing users to select fonts) and by recognizing the extensions to the ChooseFont data structures in Chicago. Even Windows–based applications—which, though not originally designed for Chicago, still support formatted text, but not multilingual messages—may gain some basic level of support for multilingual content. If an application uses the ChooseFont common dialog box, it benefits from the enhancements, allowing users to select from the full range of character sets and fonts configured in the system. As long as the application saves the complete logical font data structure representation for fonts, an existing Windows–based application can get by without being aware that the font change a user has made includes a possible change of character set. (Applications do generally save this data, at least when saving text in their native format. Fewer save this when writing to interchange formats such as RTF.)

Multilingual Support for Exchanging Information Via the ClipBoard

A good multilingual-aware application should can exchange multilingual content with other aware applications and can exchange appropriate flat text to others, within the limitations of the ASCII text formats. Chicago provides special support in the data exchange APIs to pass language information along with the rich text data.

Win32 National Language Support APIs

When they install Chicago, users specify a locale preference. (This preference can be changed later via Control Panel.) The Win32 NLS APIs can use the user' default locale settings or a specific locale setting.

By using the Win32 NLS APIs, developers can easily integrate international language support in their Win32–based applications. These APIs are supported both on the Chicago and Windows NT operating system platforms (with limited support available with Win32s under Windows 3.1) and allow applications to properly retrieve regional and language settings, format date and time, sort lists according to cultural rules, compare and map strings, and determine character type information.

This means that a developer in the U.S. can be assured that the sorting order and date formats that Microsoft provides with the operating system are correct. Therefore, the developer has only to use the appropriate Win32 NLS APIs to sort or display information.

By using the Win32 NLS APIs, developers can more easily develop applications for new global markets. Using this API set also lowers development costs by eliminating the need for proprietary sorting methods, parsing the WIN.INI file or Registry, and locale-specific coding. Perhaps more importantly for developers, the API set provides a mechanism for accurate and consistent behavior on each 32-bit Microsoft Windows platforms.

End-users benefit because the API set ensures that information is handled and displayed properly for a given locale-specific format. In addition, users don't have to worry as to whether their international text is being sorted properly.

Try It!

Mouse

To be able to examine the improvements present in Chicago to switch between language types and keyboard layouts, you've got to Try It!

Support for Multilingual Content

To demonstrate the multilingual content support in Chicago, try these simple procedures to install different language support.

- In Control Panel, open the Keyboard icon and click on the Language tab. Add a couple of keyboards (for example, Swedish and French) and then choose OK. On the right corner of the Chicago Taskbar, notice a small square is displayed in the status area to represent the active keyboard layout—two letters are displayed in the square to represent the language, for instance EN for English.

- Start a word processing application and try to create a document in which to try the multilingual content support. To hot-switch between different input languages, press ALT+SHIFT, and toggle through the available configured languages. Once you have switched to a new input language, type something (a multilingual-aware application would automatically switch the proper font). Of course you have to know where the keys are on that country-specific keyboard layout.

In a true multilingual-aware application, you can scroll the text and the application will automatically switch the current input language to match the proper language format when you scroll through different languages in the text.

Accessibility

Microsoft is committed to making computers easier to use for everyone, including individuals with disabilities. Personal computers are powerful tools that enable people to work, create, and communicate in ways that might otherwise be difficult or impossible. This vision of enabling all people can be realized only if individuals with disabilities have equal access to the powerful world of personal computing.

The issue of computer accessibility in the home and workplace for people with disabilities is becoming increasingly important. Seven to nine out of every ten major corporations employ people with disabilities who may need to use computers as part of their jobs. In the U.S. alone, an estimated 30+ million people have disabilities that can potentially limit their ability to use computers. Additionally, as the population ages, even more people will experience functional limitations, causing the issue of computer accessibility to become even more important to the population as a whole.

Legislation, such as the Americans with Disabilities Act (which affects private businesses with more than 15 employees) and Section 508 of the Rehabilitation Act (which addresses government spending), also brings accessibility issues to the forefront in both the public and private sectors.

Microsoft already offers a number of products specifically for users with disabilities and includes features in our mainstream software products that help make them more accessible. Our two most prominent accessibility products are Access Pack for Microsoft Windows and AccessDOS. Both were developed by the Trace Research and Development Center at the University of Wisconsin-Madison, using research funded by NIDRR. These products enhance the Windows and MS-DOS operating systems by adding a variety of features which make the computer more accessible for users with limited dexterity or hearing impairments. Microsoft distributes these utilities at no charge to the customer, and documents their availability with each of our new products.

Chicago offers several enhancements designed to make the system more accessible and easier to use for individuals with disabilities. In recent years Microsoft has established close relationships with users who have disabilities, organizations representing disabled individuals, workers in the rehabilitation field, and software developers who create products for this market. Based on their combined input, Microsoft has defined specific design goals for Chicago:

- Integrate and improve the features from Access Pack which compensate for difficulties some individuals have using the keyboard or the mouse
- Make the visual user interface easier to customize for people with limited vision
- Provide additional visual feedback for users are deaf or hard-of-hearing

- Provide new API and “hooks” for Independent Software Vendors (ISVs) developing third-party accessibility aids, including those which allow blind individuals to use Windows
- Make information on accessibility solutions more widely available and increase public awareness of these issues

Throughout the Chicago product, enhancements designed to meet these goals are included. This section of the *Chicago Reviewer’s Guide* describes the enhancements to the product which will make computing easier for individuals who have disabilities.

Summary of Improvements Over Windows 3.1

The primary improvements in accessibility for Chicago are:

- Scaleable user-interface elements
- Compensate for difficulties using the keyboard
- Keyboard emulation of the mouse
- Support alternative input devices which emulate the keyboard and mouse
- Provide visual cues to tell the user when the application is making sounds
- Advise applications when the user has limited vision
- Advise applications when the user needs additional keyboard support due to difficulty using a mouse
- Provide new API for accessibility software
- Optimize keyboard layouts for users who type with a single hand, a single finger or a mouthstick
- Include audible prompts during Setup for users who have low vision
- Color schemes which are optimized for users with low vision
- Include accessibility information in our documentation

General Accessibility Enhancement Features

Online Help

An Accessibility section in the Chicago help contents and index provides a quick reference and pointer to topics that can help adjust the behavior of the system for people with disabilities.

Controlling the Accessibility Features

Most of the accessibility features described in this section are adjusted through the Chicago Control Panel. This allows you to turn the features on or off and to customize timings, feedback, and other behavior for your own particular needs.

Note In this Beta-1 release of Chicago, the Control Panel property sheets for accessibility are not yet available. Instead, a separate Access Utility is provided in the Chicago directory, ACCESS.EXE, to perform these tasks.

Emergency Hotkeys

Most of the accessibility features described in this section are adjusted through the Control Panel. But if a user is unable to use the computer until an accessibility feature is turned on, how can they use Control Panel to activate it? This chicken-and-the-egg problem is solved by providing emergency hotkeys which by which a user can temporarily turn on the specific feature he or she needs. Then, once a feature is turned on, the user can navigate to Control Panel and adjust the feature to his or her own preferences, or turn it on permanently.

The same hot-key can use used to temporarily turn off one of the features, if it gets in the way or to enable another person to use the computer.

We have been worked hard to make sure that the emergency hotkeys will not get in the way of users who don't need them. Each hot-key is designed to be obscure key combinations or key sequences which should not conflict with applications. If such a conflict does arise, the hotkeys can be disabled, and the user can still use the feature or not as needed.

As an additional precaution, each emergency hot-key plays a rising tone, and also brings up a confirmation dialog box that briefly explains the feature and how it was activated. If the user pressed the hot-key unintentionally, this allows the user to cancel the feature's activation. It also provides a quick path to more detailed Help and the Control Panel settings for that feature, allowing the user to disable the hot-key permanently.

Accessibility TimeOut

The Accessibility TimeOut turns off Access Pack's functionality after the system has been idle for a certain period of time. It returns the system to its default configuration. This feature is useful on machines shared by multiple users.

The Accessibility TimeOut can be adjusted using Control Panel.

Note In the Chicago Beta-1 release this is available through the separate ACCESS.EXE utility in the Chicago directory.

Accessibility Status Indicator

An optional visual indicator is provided that tells the user which accessibility features are turned on, helping users unfamiliar with the features identify the cause of unfamiliar behavior. The indicator also provides feedback on the keys and mouse buttons currently being “held down” by the StickyKeys and MouseKeys features.

Features for Users with Low Vision

Scaleable User Interface Elements

Users who have limited vision or who suffer eyestrain during normal use of Windows can now adjust the sizes of window titles, scroll bars, borders, menu text, and other standard screen elements. These sizes are completely customizable through the Chicago Control Panel. You can also choose between two sizes for the built-in system font.

Customizable Mouse Pointer

Users who have difficulty seeing or following the mouse pointer can now choose between three sizes: normal, large, and extra large. In coming releases, you will also be able to adjust the color or add animation, both of which can increase the pointer’s visibility.

High-Contrast Color Schemes

The Windows color schemes allows users to choose from several well-designed sets of screen-color options designed both to match users’ individual tastes and to meet their visual needs. Chicago’s new color schemes include high-contrast colors designed to optimize the visibility of screen objects, making it easier for users with visual impairments.

High-Contrast Mode

Many users with low vision require high contrast between foreground and background objects to be able to distinguish one from the other. For example, they may not be able to easily read black text on a gray background, or text drawn over a picture. Users can now set a global flag to advise Chicago and applications that they need information presented with high contrast.

Chicago also provides an emergency hot-key which allows a person to set the computer into high-contrast mode when they may be unable to use Control Panel, or when the current color scheme makes the computer unusable for them. This hot-key allows them to choose an alternate color scheme which better meets their needs.

High-Contrast Mode can be turned on or off using an emergency hot-key, by pressing left ALT, left SHIFT, and PRINT SCREEN keys simultaneously.

Features for Making Keyboard and Mouse Input Easier

StickyKeys

Many software programs require the user to press two or three keys at one time. For people who type with a single finger or a mouthstick, that just isn't possible. StickyKeys allows users to press one key at a time and instructs Windows to respond as if they had been pressed simultaneously.

When StickyKeys is on, pressing any modifier key (that is, CTRL, ALT, or SHIFT) will latch that key down until you release the mouse button or a non-modifier key. Pressing a modifier key twice in a row will lock it down until it is tapped a third time.

In this Beta-1 release StickyKeys functionality is adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release), or it can be turned on or off using an emergency hot-key, by pressing the SHIFT key five consecutive times.

SlowKeys

The sensitivity of the keyboard can be a major problem for some individuals, especially if they often press keys accidentally. SlowKeys instructs Windows to disregard keystrokes that are not held down for a minimum period of time. This allows a users to brush against keys without any ill effect, and when the user get a finger on the proper key, the user can hold the key down until the character prints to the screen.

In this Beta-1 release SlowKeys functionality is adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release), or it can be turned on or off using an emergency hot-key, by holding down the right SHIFT key for eight seconds. (This hot-key also turns on RepeatKeys.)

RepeatKeys

Most keyboards allow users to repeat a key just by holding it down. This feature is convenient for some, but can be a major annoyance for people who can't lift their fingers off the keyboard quickly. RepeatKeys lets users adjust the repeat rate or disable it altogether.

In this Beta-1 release RepeatKeys is adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release), or it can be turned on or off using an emergency hot-key, by holding down the right SHIFT key for eight seconds (This hot-key also turns on SlowKeys.)

BounceKeys

For users who "bounce" keys, resulting in double strokes of the same key or other similar errors, BounceKeys instructs Windows to ignore unintended keystrokes.

In this Beta-1 release BounceKeys is adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release), or it can be turned on or off using an emergency hot-key, by holding down the right SHIFT key for twelve seconds. You will hear an up-siren after eight seconds, and another double-tone after twelve seconds. Releasing the SHIFT key after the double-tone will activate BounceKeys.

MouseKeys

This feature lets individuals control the mouse pointer using the keyboard. Chicago is designed to allow the user to perform all actions without needing a mouse, but some applications may require one, and a mouse may be more convenient for some tasks. MouseKeys is also useful for graphic artists and others who need to position the pointer with great accuracy. You do not need to have a mouse to use this feature.

When MouseKeys is on, use the following keys to navigate the pointer on your screen:

- Press any number key except 5 (these are also called the direction keys) on the numeric keypad to move the pointer in the direction indicated by the diagram below.
- Use the 5 key for a single mouse-button click and the + key for a double-click.
- To drag and release an object, with the pointer on the object, press INS to begin dragging, then move the object to its new location and DEL to release it.
- Select the left, right, or both mouse buttons for clicking by pressing the /, -, or * key, respectively.
- Hold down the CTRL key while using the direction keys (numeric keys, except for 5) to "jump" the pointer in large increments across the screen.
- Hold down the SHIFT key while using the direction keys to move the mouse a single pixel at a time for greater accuracy.

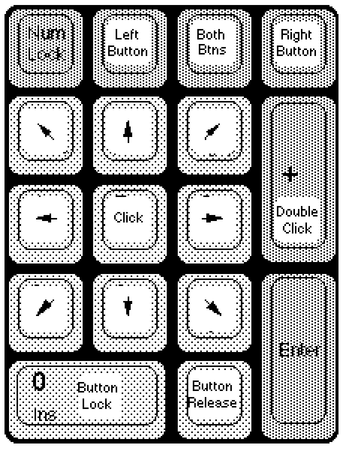


Figure 94. Keys on the numeric keypad that control the mouse pointer

In this Beta-1 release MouseKeys can be adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release), or it can be turned on or off using an emergency hot-key, by pressing the left ALT, left SHIFT and NUM LOCK keys simultaneously.

ToggleKeys

ToggleKeys provide audio cues—high and low beeps—to tell the user whether a toggle key is active or inactive. It applies to the CAPS LOCK, NUM LOCK, and SCROLL LOCK keys.

In this Beta-1 release ToggleKeys can be adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release), or it can be turned on or off using an emergency hot-key, by holding down the NUM LOCK key for eight seconds.

Features for Users Who Are Hearing-Impaired

ShowSounds

Some applications present information audibly, as wave-files containing digitized speech or through audible cues that each convey a different meaning. These cues might be unusable by a person who is deaf or hard-of-hearing, someone who works in a very noisy environment, or someone who turns off the computer's speakers in a very quiet work environment. In Chicago, you can set a global flag to let applications know you want visible feedback—in effect asking the applications to be "close captioned.

ShowSounds is adjusted using Control Panel. (In the Chicago Beta-1 this feature can be adjusted using the separate ACCESS.EXE utility instead of Control Panel.)

SoundSentry

SoundSentry tells Windows to send a visual cue, such as a blinking title bar or screen flash whenever there is a system beep. This allows users to see the message that may not have been heard.

In this Beta-1 release SoundSentry is adjusted using the ACCESS.EXE utility (which will be integrated into Control Panel in the next Beta release).

Support for Alternative Input Devices

SerialKeys

This feature, in conjunction with a communications aid interface device, allows the user to control the computer using an alternative input device. Such a device needs only to send coded command strings through the computer's serial port to specify keystrokes and mouse events which are then treated like normal keyboard and mouse input.

Support for Multiple Pointing Devices

Chicago's new Plug and Play architecture inherently supports multiple pointing devices all cooperating together. This allows seamless addition of alternative pointing devices, such as head-pointers or eye-gaze systems without the need to replace or disable the normal mouse.

Features for Software Developers

Accessibility Guidelines for Software Developers

Chicago contains many built-in features designed to make the computer more accessible to people with disabilities. To make a computer running Chicago truly accessible, application developers must provide access to their applications' features, taking care to avoid incompatibilities with accessibility aids.

As part of the *Chicago Software Development Kit* and *Chicago UI Design Guidelines*, Microsoft has provided developers with documentation which not only outlines to these important concepts, but provides technical and design tips to help ISVs produce more accessible applications. Most of these tips will mean very little additional work to the designer, as long as the application designer is aware of the issues and incorporates accessibility into the application design at an early stage. By providing this information to application developers, Microsoft hopes to increase the general level of accessibility of all software running on the Windows platform.

Methods for Simulating Input

Chicago now allows developers of voice-input systems and other alternative input systems to easily simulate keyboard and mouse input using fully documented and supported procedures.

Chaining Display Drivers

Some accessibility aids, such as screen review packages for low-vision users, need to detect information as it is drawn to the screen. Chicago supports chaining display drivers that allow these utilities to intercept text and graphics being drawn, without interfering with the normal computer operation.

New Common Controls

Many accessibility aids have difficulty working with applications which implement non-standard controls. Chicago introduces a whole new set of controls available for mainstream software developers, and these standardized implementations are designed to cooperate with accessibility aids.

Try It!

To see how these accessibility features in Chicago make it easy to customize the appearance and behavior of your computer, you can try them out yourself!

Don’t Touch that Mouse

Press the left ALT, the left SHIFT and the NUM LOCK keys simultaneously, and you’ll be able to drag-and-drop, and click or double-click both the primary and secondary mouse buttons simply using your keyboard’s numeric keypad. For more information, see the “MouseKeys” section above.

Try Typing With a Pencil

Suppose you could only type with a single finger, or with a stick held between your teeth. How would you press ALT+ TAB? Press a SHIFT key consecutive five times to try out StickyKeys. Once it’s activated, press the ALT key and see what happens. Press TAB and you’ll have just typed two keys at once with a single finger. Press the ALT key twice, then press TAB a few times to see the ALT+ TAB window and cycle through all the tasks you have running. When you’re satisfied, press ALT one more time to release it. When you’re ready to move on you can turn off this feature just by pressing two keys at the same time.

Support for MS-DOS–based Applications

All of the accessibility features described here are available even when you're running MS-DOS–based applications. Start an MS-DOS application and try StickyKeys or MouseKeys—these features are available any time you need them, whatever you may be doing.

What Makes A Great Chicago Application?

While Chicago provides compatibility for running MS-DOS and Win16–based applications, users benefit from additional functionality supported by Win32–based applications. Win32–based applications benefit from the preemptive multitasking architecture of Chicago and the increased robustness and protection for running applications. In addition, there are six key areas that make a great Chicago application from the users perspective:

- The Win32 Application Programming Interface (API)
- Object Linking and Embedding (OLE) 2.0 functionality
- Windows User Interface Style Guideline 4.0
- Support for handling Plug and Play events
- Support for manipulating long filenames
- Adherence to common setup guidelines for consistent software installation

In the next section discusses why these components make these applications great for users.

The Win32 Application Programming Interface

Microsoft supports the use of the Win32 API on three operating system platforms: Windows NT, Windows "Chicago", and Windows 3.1 with Win32s. Each operating system supports a common set of Win32 APIs, allowing applications to be developed for a single API set and run on multiple platforms. This allows application developers and corporate developers to learn a single API set, and leverage development resources to support a broad base of hardware systems. Users benefit from being able to run the same application on multiple platforms, and increased system reliability under Chicago due to improved robustness and memory protection available to 32-bit applications.

Chicago delivers a robust and powerful 32-bit platform. 32-bit applications for Chicago are preemptively multitasked, run in private address spaces, and can spawn multiple threads of execution. Preemptive multitasking ensures excellent system responsiveness, allowing users to run multiple applications simultaneously and integrate personal productivity and business-critical applications in a smooth manner. This is similar to the model that Windows NT uses today. The use of a private address space for each Win32–based application ensures that multiple applications can run simultaneously without interfering with each other or the operating system itself. The Chicago operating system provides smooth, preemptive multitasking and protected virtual memory, because Chicago is based on a re-architected 32-bit protected-mode kernel and a 32-bit protected-mode driver model.

User Benefits of Using Win32-based Applications

Running 32-bit applications under Chicago provides the following improvements from an end-user perspective:

- Background multitasking for running multiple applications that is smoother because of Chicago's preemptive multitasking architecture
- Overall system performance improvements due to 32-bit operating system components
- Robustness and system reliability improvements due to 32-bit memory protection and separate message queues
- New applications functionality because of Win32 and other operating system services
- File manipulation that is easier because of Chicago's long file name support

OLE 2.0 Functionality

Users are getting and using more applications per PC than ever before. In 1992, InfoCorp reported the average number of applications purchased per desktop running the Windows operating system increased to more than 7 programs, up from an average of 3.4 programs for customers using the MS-DOS® operating system in 1986. People are not just acquiring more applications, they are using them together. Research shows that users cite the ability to move and share information among applications as the most important reason for using windows applications.

Users who learn one Windows application find it easy to learn a second or third. So, as users access several applications in the course of creating a compound document, they'll feel comfortable with those applications.

The Solution for Application Integration

Object Linking and Embedding (OLE) 2.0 is a mechanism that allows applications to interoperate more effectively, thereby allowing users to work more productively. Users of OLE 2.0 applications create and manage compound documents. These are documents which seamlessly incorporate data, or objects, of different formats. Sound clips, spreadsheets, text, bitmaps are some examples of objects commonly found in compound documents. Each object is created and maintained by its server application. But through the use of OLE 2.0, the services of the different server applications are integrated. Users feel as if a single application, with all the functionality of each of the server applications, is being used. Users of OLE 2.0 enabled applications don't need to be concerned with managing and switching between the various server applications; they focus solely on the compound document and the task being performed OLE 2.0-based features.

Features of OLE 2.0

With OLE 2.0, Microsoft Chicago increases the degree of application integration available to any applications which take advantage of the services. This gives users tangible benefits, allowing them to share data and functionality across applications, and to combine them as they please. Because OLE 2.0 is based on an open industry-standard, users can extend their applications with additional third-party products, further expanding their choice and flexibility.

OLE 2.0 provides the following features to allow users to easily integrate information into multiple applications:

- **Cross-application drag-and-drop**

  Users can drag-and-drop graphs, tables, and pictures directly onto slides, worksheets, and documents to mix text, data, and graphics into compound documents. Using drag-and-drop is faster and more intuitive than using the Clipboard to cut and paste.

- **Visual editing**

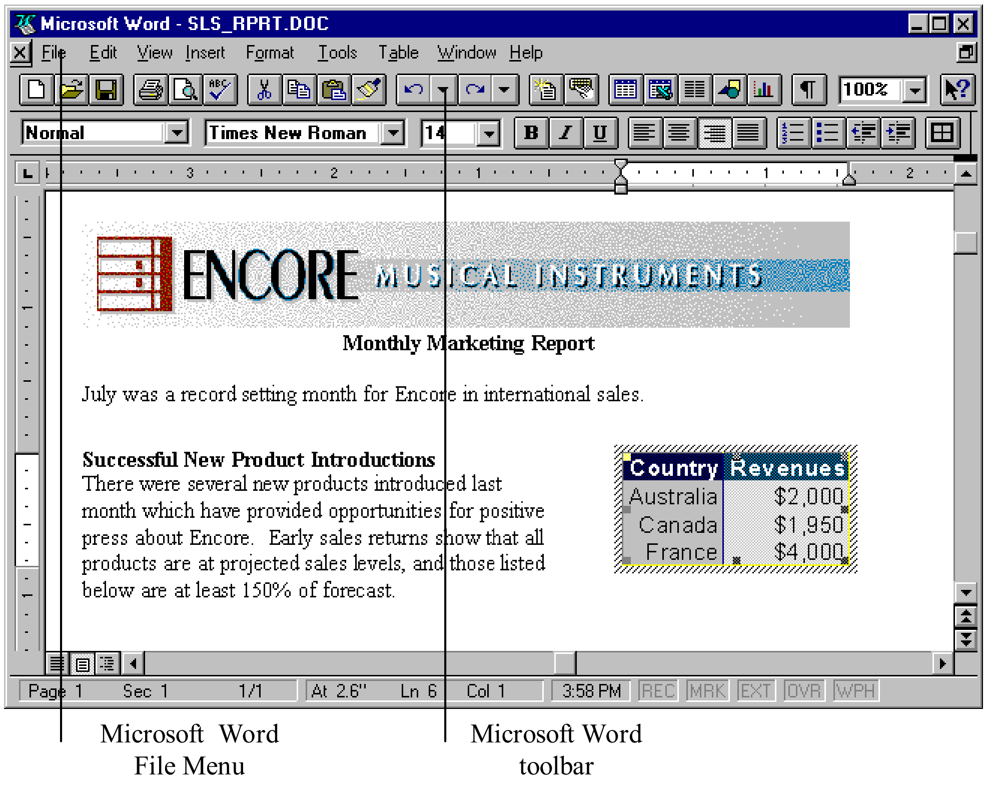
  With visual editing, the user can double-click an object to directly edit it while remaining in the original document. For instance, double-click an embedded Microsoft Excel worksheet in a Word document, and the Microsoft Excel menus and toolbars automatically appear within the context of Word. Unlike OLE 1.0, the user is not launched into a separate Microsoft Excel window to work on the spreadsheet data.

Drag-and-Drop

Drag-and-drop is a new and more intuitive way to move data between applications. The most widely used way to transfer data between applications has been to use the Clipboard. But this involves multiple steps; namely using the Copy operation, moving to the destination application, and using the Paste command. A more effective way to move information—drag-and-drop—already exists within applications and, with OLE 2.0, it now works between applications, too. The user simply selects an object in one application, drags it to its destination in another application and drops it into place. Objects also can be dragged over the desktop to system resource icons such as printers and mailboxes, making it faster and easier to send, print, or share files.

Visual Editing

Visual editing makes revising a compound document faster, easier, and more intuitive. For example, a Microsoft Excel worksheet that's contained within a Word document (see Figure 96 below), can be double-clicked by the user. The user then is able to interact with the Microsoft Excel worksheet right there, without switching to a different application or window.


Microsoft Word - SLS_RPRT.DOC
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
14
100%
Page 1 Sec 1 1/1 At 2.6" Ln 6 Col 1 3:58 PM REC MRK EXT OVR WPH
Microsoft Word File Menu
Microsoft Word toolbar

ENCORE MUSICAL INSTRUMENTS

Monthly Marketing Report

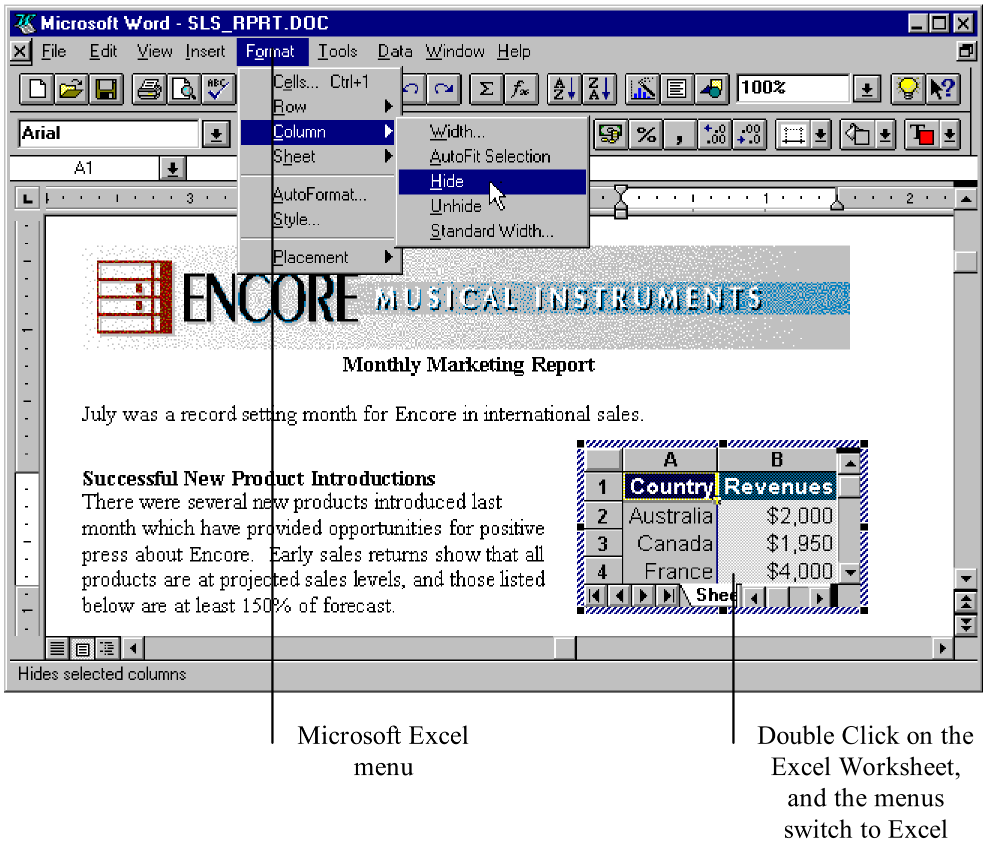
July was a record setting month for Encore in international sales.

Successful New Product Introductions
There were several new products introduced last month which have provided opportunities for positive press about Encore. Early sales returns show that all products are at projected sales levels, and those listed below are at least 150% of forecast.

Country	Revenues
Australia	$2,000
Canada	$1,950
France	$4,000

Figure 95. Sample OLE 2 Spreadsheet Object Embedded in a Word Processing Document

The menus and toolbars necessary to interact with the Microsoft Excel spreadsheet temporarily replace the existing menus and controls of Word. Microsoft Excel, the application that is needed to edit or modify the spreadsheet, partially "takes over" the Word document window (see Figure 97 below). When the user wants to work on the word processing portion of the document, the focus returns to Word, and the original Word menus and controls are restored.


Microsoft Word - SLS_RPRT.DOC
File Edit View Insert Format Tools Data Window Help
Cells... Ctrl+1
Row
Column
Sheet
AutoFormat...
Style...
Placement
Width...
AutoFit Selection
Hide
Unhide
Standard Width...
Arial
A1
100%
Hides selected columns
ENCORE MUSICAL INSTRUMENTS
Microsoft Excel menu
Double Click on the Excel Worksheet, and the menus switch to Excel

Monthly Marketing Report

July was a record setting month for Encore in international sales.

Successful New Product Introductions
There were several new products introduced last month which have provided opportunities for positive press about Encore. Early sales returns show that all products are at projected sales levels, and those listed below are at least 150% of forecast.

	A	B
1	Country	Revenues
2	Australia	$2,000
3	Canada	$1,950
4	France	$4,000

Figure 96. Activating OLE 2 Object Changes Menu Context

The advantage of visual editing is even greater for example, when users create compound Word documents including large numbers of objects created by different applications, such as Microsoft Excel worksheets and charts, PowerPoint graphics, sound and video clips, and so on. Instead of switching back and forth among different windows to update the objects, the user is presented with a single document window in Word, providing a single location for editing and other interactions with the data. Visual editing offers users a more “document-centric” approach, putting the primary focus on the creation and manipulation of information rather than on the operation of the environment and its applications.

Windows User Interface Style Guideline 4.0

As in previous version of Windows, one of the reasons why applications are easy to learn is the fact that they look and act alike. With Chicago, Microsoft has taken great steps to improve the basic common controls that all applications can share. These controls have evolved based on user feedback and extensive usability testing here at Microsoft. Applications that use these controls provide the users with commonality and improved features—such as being able to create new folders in the Save As dialog box—without having to switch to the Explorer or File Manager.

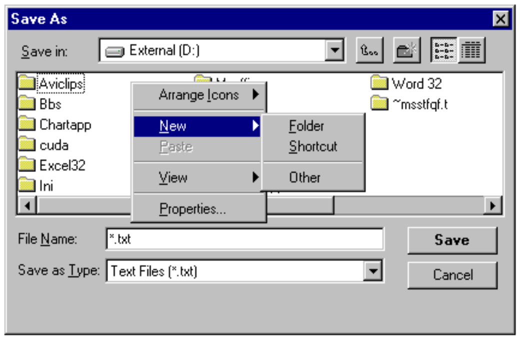


Figure 97. Save As Common Dialog

In the new Printer Properties dialog you can see examples of some of the new controls with make access to features even easier for users. At the top of the dialog you can see tabs for “Paper,” “Device Options,” Graphics,” and, in the case of the printer shown in Figure 99, “Postscript.” Clicking any tab presents the user with properties for that particular area. Another new control available to all applications is the Spin Control next to the number of Copies.

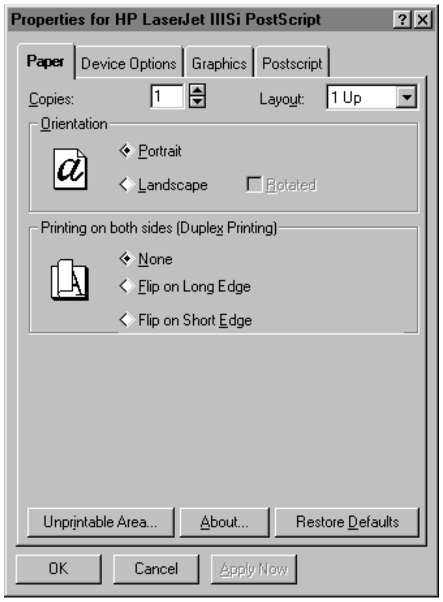
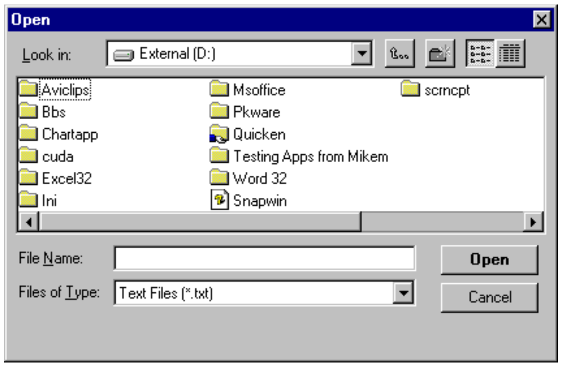


Figure 98. Sample Tabbed Dialog Box Property Sheet

The new Open Dialog allows the user to see long filenames, and navigate the entire PC, and connected network to look for files to open.



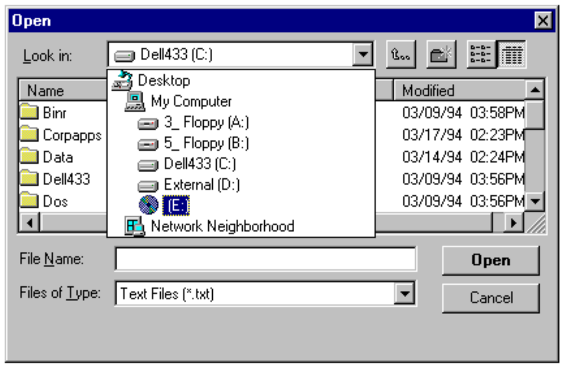
Look in: External (D:)

File Name:

Files of Type: Text Files (*.txt)

Figure 99. Open Common Dialog

The new Open dialog also uses Tree Lists to allow the user to navigate through the hierarchy of the hard disks attached to the computer and through the network to which the computer is connected.



Look in: Dell433 (C:)

File Name:

Files of Type: Text Files (*.txt)

Figure 100. Open Common Dialog Provides Easy Access to Network Resources

Figure 102 shows another new control that makes viewing and accessing hierarchical information even easier. This is a tree list control found in the property sheet for the Device Manager in the System section of Control Panel. As users expand and collapse the tree, they can see information relevant to their topic of choice.

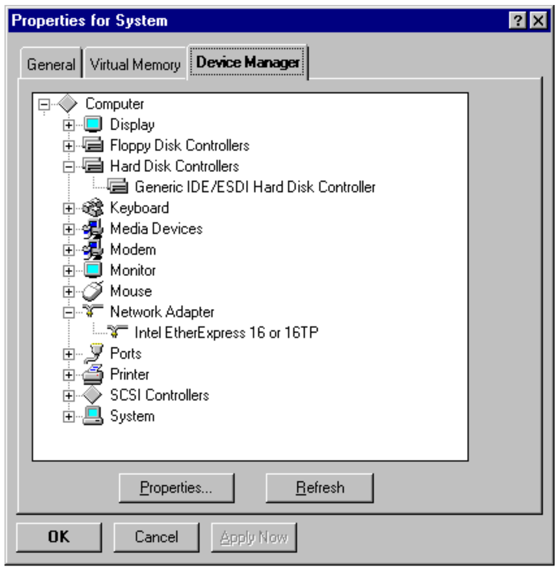


Figure 101. Sample Tree List Control

Applications no longer have to include their own custom slider controls. Figure 103 shows the new common slider control included in Chicago.

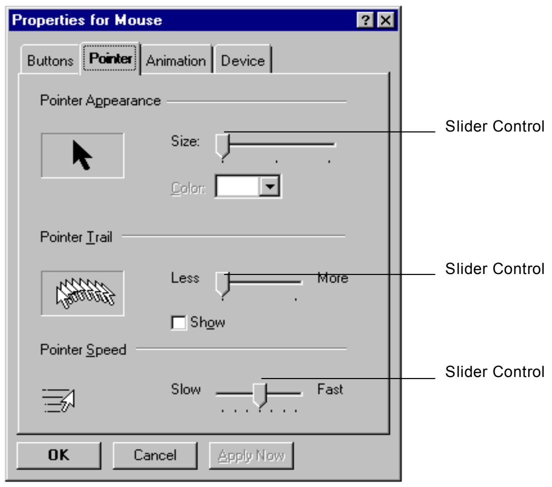


Figure 102. Sample Slider Control

There are many new common controls, tool bar, status bar, column heading, tabs, slider, progress indicator, rich text control, list view, tree view... and much more. Great Chicago applications will use these new controls to make the users access consistent across applications and make the entire system much easier to use.

Support for Plug and Play Events

Applications that provide Plug and Play event awareness help users by seamlessly adapting to hardware configuration changes. Users will reap two key benefits:

- **Applications automatically recognize and respond to hardware changes**

  Consider this scenario: Chicago's desktop Control Panel recognizes changes in the video resolution and configures the computer accordingly. Now suppose a user who has a mobile PC installed in a docking station and who is using an external monitor running in 1024 x 768 resolution mode. When the user undocks the PC, the desktop Control Panel recognizes this action and seamlessly switches resolution for the mobile PC to 640 x 480. When this change occurs, Plug and Play aware applications will resize their windows and toolbars accordingly. The user doesn't have to do a thing; it's all automatic.

  Here's another scenario: Suppose you were using a mobile PC to work on a document. One of the most critical situations you could find yourself in is running out of battery power. With Chicago, your computer sends a message to all the active Plug and Play aware applications, telling them to shut down and save your data because the power is going off in a few minutes.

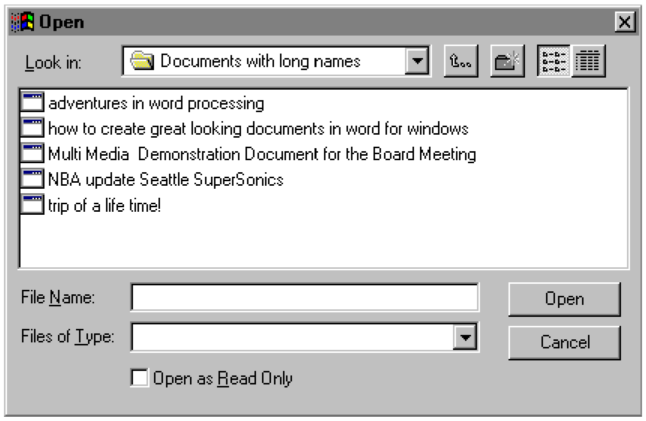
- **Applications warn users about open network files when hot-undocking their computers**

  Suppose that you have a PCMCIA network card installed in your laptop computer, and you are leaving the office. As you leave the office, you switch PCMCIA cards and install your modem for remote access. With Plug and Play, you don't have to fuss with software configuration—you're set. Chicago simply knows that the network has gone away, and that the network card is now replaced by a modem. A Plug and Play aware Email application also learns this information from the operating system. It knows that there is no more network connection and it now needs to use the modem to make connections. The software configurations are automatically made for you.

It should be quite clear that applications which are Plug and Play aware provide seamless adaptation to changes in the hardware configuration. With applications that are Plug and Play aware, users can focus on their work, not their configuration.

Long Filename Support

As you have seen by now, there is a much improved new shell for Chicago. But the shell itself is only part of what is really here for users. Now an application that takes advantage of this new shell support can offer their users long filenames and direct file viewing. Long filename support means that documents no longer have to be limited to eight characters for names. They now can have up to 255 characters. Instead of "23ISM_JB.doc", you can name a file "Status report July 23 regarding the ISM project for my boss Jim Bernstein"—a title that really tells you what the document is about. Applications that support the Viewer capabilities in Chicago provide users with a quick and easy way to view their documents directly from the shell without launching the application.



adventures in word processing
how to create great looking documents in word for windows
Multi Media Demonstration Document for the Board Meeting
NBA update Seattle SuperSonics
trip of a life time!

Figure 103. Chicago Applications Support Long Filenames

Consistent Setup Guidelines

In the past users have generally had an easy way to setup their new applications, but removing these applications from their hard disks was not so simple. Most Windows 3.1 users eventually ended up with all kinds of files on their systems which were never used because they belonged to some previously-deleted application. And since many applications use the same library files, with Windows 3.1 it was quite common to have several copies of a files installed in different places on the hard disk, an inefficient use of precious disk space. Another common problem with Windows 3.1 is “files, files, files, everywhere”—applications put files in their own directory, in the Windows directory, in the System directory, in the root, you name it. This creates a real mess when trying to keep track of what’s where.

The *Chicago Software Development Kit* (SDK) offers to developers some guidelines for consistent installation locations and uninstall functionality in their applications. Common libraries can be shared by applications thereby reducing the amount of disk space consumed by duplication. The guidelines also set standards for where developers should put all their files on the hard disk. No more “files, files everywhere” syndrome. These guidelines provide a much easier, powerful, and compatible structure for users. Setup programs that follow the guidelines will all operate similarly, use consistent naming conventions, and offer the same setup options—thus reducing the learning curve for users, and improved manageability and support for corporations allowing for increased efficiency in remote administration installed software.

Chicago Questions and Answers

May 1994

Microsoft is continually enhancing its Windows operating system product line to deliver easy to use, yet powerful products that exploit the latest advances in microcomputer hardware technology. There is a great deal of interest in and speculation about the "Chicago" project, the technology development effort which will deliver the next major release of Windows for the mainstream desktop and portable PC. The purpose of these information is to address the most common questions that customers have voiced about Chicago.

What is Chicago?

- **What is Chicago and how does it compare to the Microsoft® Windows™ 3.1, Windows™ for Workgroups and Windows NT™ operating systems?**

"Chicago" is the code name for a development project that will produce the successor to Windows 3.*x* and Windows for Workgroups 3.*x*. The Chicago project encompasses multiple technologies that will make everything customers want to do on mainstream PCs running Windows even easier. Chicago will deliver faster multitasking performance and will provide the highest level of compatibility with the applications and devices customers use today. Chicago will be the right choice for customers who primarily want to run business and personal productivity applications, and for use on home computers to run recreational and educational applications. Windows NT is designed for the most demanding business uses such as developer workstations, or advanced engineering and financial workstations. Windows NT is the right choice for customers who need the highest level of protection for their data and applications, and scalability to multiprocessor and RISC systems.

- **What are the key benefits and features of Chicago? What features will Chicago not have?**

For customers, Chicago will present a major step forward in functionality on mainstream desktop and portable PC platforms by providing a system that is even easier, faster, and more powerful to use, and that is compatible with the applications and devices in which customers have already invested.

Ease of use will be improved through the Plug and Play architecture and an improved, more intuitive user interface. With the introduction of Chicago, Windows' "engine" is being revamped to improve performance and provide smooth, responsive multitasking. Chicago will be a complete, integrated protected-mode operating system that does not require or use a separate version of MS-DOS. It implements the Win32® API and provides preemptive multitasking and multiple threads of execution for 32-bit applications. Chicago will include reliable and open networking support and high-performance, as well as messaging (email and fax) and remote access services.

To be the successor to Windows 3.*x* and Windows for Workgroups 3.*x*, Chicago will meet a number of key customer requirements. First, Chicago will be compatible with most all current applications and drivers for MS-DOS and Windows (shell utilities and file management utilities will need to be revised to be compatible with Chicago's new components). When a customer upgrades to Chicago, performance will meet or exceed performance of Windows 3.1, as long as the customer has an 80386 or higher system with at least 4MB of RAM. For systems with more memory, performance will be significantly improved over Windows 3.1. The transition to the new user interface will be easy for existing Windows users, and companies who want to make the transition at their own pace will still be able to run Program Manager and File Manager during the transition period. The Setup program will provide a very simple safe installation process for end-users, and will provide tools for system administrators to customize the configuration of Chicago.

Chicago will *not* be processor-independent, nor will it support symmetric multiprocessing systems, nor provide C2-level security. These features cannot be delivered on the mainstream platform while still meeting the performance, compatibility, and memory targets necessary to create a compelling upgrade for the huge installed base of Windows users. If these features are important to a customer, Windows NT is the right product to use.

Why Will I Want Chicago?

■ **Why will individual customers want to upgrade to Chicago?**

The sheer quantity of improvements to Windows in Chicago represents a great value for customers— there's something in the product for everyone. Top on the list of requested improvements is an easier way to work with the PC. Chicago's **new user interface** will make everything customers do even easier. This is true for the less-experienced person who finds it confusing to manage multiple windows on the screen and difficult to learn different tools for managing programs, files and printers. It's also true for the experienced person who craves greater efficiency and flexibility.

Long filename support is just one of the many usability improvements in Chicago. But improving ease of use goes beyond fixing problems with Windows—it encompasses the hardware, applications and network as well. **Plug and Play** will make hardware setup automatic, and **built-in networking** will make installing a network as easy as setting up a printer.

The other major improvement customers are asking for is greater efficiency and power. They want to get their work done faster. They want to run more than one application or task at the same time so they don't have to wait for their PC. They want to use their computer to access files, electronic mail, and public information networks from any location—at work, at home, or on the road. They want better multimedia, whether for those addictive MS-DOS-based games or for TV-quality video resolution for teleconferencing. Some of the highlights that address these requests:

- **Preemptive multitasking**. Chicago can smoothly and responsively multitask 32-bit applications.
- **Scaleable performance**. Chicago performance increases more rapidly than Windows 3.1 as the amount of RAM increases, due to its high performance 32-bit architecture.
- **Support for 32-bit applications**. Chicago's support for the Win32 API means that customers can look forward to a new generation of easier, faster, and more reliable applications.
- **Increased reliability**. Chicago increases protection for existing MS-DOS and Windows applications, and provides the highest level of protection for new 32-bit Windows applications.
- **Faster printing**. Chicago's new 32-bit printing subsystem means a lot less time waiting for print jobs to finish and better system response when jobs are printing in the background.

- **Better multimedia**. Just as Windows 3.1 made sound a part of the system, Chicago now includes video playback support. The video system and CD-ROM file system (CDFS) will provide high-quality output for multimedia applications.
- **More memory for MS-DOS applications**. Chicago's use of protected-mode drivers means customers will have more than 600K free conventional memory in each MS-DOS session—even when they are connected to the network and using a CD-ROM drive and a mouse.
- **Remote networking**. Chicago provides a remote networking client that enables dial-in to multiple networks (including the Internet) and a remote networking server that lets any Chicago PC act as a secure, single-line dial-in network gateway.

■ **Why will companies want to upgrade to Chicago?**

MIS organizations will want to upgrade to Chicago both for all the benefits that Chicago provides their end-users, and because Chicago will **reduce support costs and increase control** over the desktop. The popularity of Windows created a number of security, reliability and management problems that consume MIS resources. Chicago will address these problems through the following:

- **Built-in, robust networking**. Whether you're running NetWare or Microsoft networks; with NetBEUI, IPX/SPX or TCP/IP; and using NDIS or ODI, Chicago has integrated support for your network client, protocol, and driver. Additional networks are added easily. Chicago includes 32-bit clients for both NetWare and Microsoft networks that are fast, reliable, and that require no conventional memory.
- **Multiple network support**. A Chicago PC can have multiple network clients and transport protocols running simultaneously for heterogeneous connectivity.
- **System management support**. The Windows Registry provides the foundation for managing the PC. the Registry holds all the pertinent information about the system—hardware, software, user preferences and privileges—and provides access to its contents over the network through a variety of industry standard interfaces, including SNMP, DMI, and Remote Procedure Call.
- **User profiles**. Chicago has the ability to present a different configuration depending on who has logged into the PC. For example, a network administrator can use this capability to prevent Joe User from deleting icons no matter what PC he uses, or to enforce a requirement that everyone in a particular department change their passwords every 90 days.
- **Network backup agents**. Chicago includes backup agents that work with the leading server-based backup products.

Ship Dates and Packaging Plans

■ **When will Chicago ship?**

Microsoft's commitment is to ship a great product, and we fully expect to ship by the end of 1994.

- **What is Daytona? When will it ship?**

  Daytona is an interim release of Windows NT focused on size and performance improvements plus enhanced connectivity. Daytona is scheduled to ship during the summer of 1994.

- **What different packages will you have for Chicago?**

  Packaging decisions will be made later in the development cycle based on customer and business needs.

- **I keep hearing rumors that Microsoft is working on versions of Chicago for non-Intel microprocessors. Is this true?**

  No, Microsoft is not working on versions of Chicago for non-Intel microprocessors. Windows NT is Microsoft's portable operating system, and it's already available on high-end Intel, MIPS®, Alpha, PowerPC, and Clipper™ computers.

- **Since the term Chicago is a code name, what will you call the product(s) that you will eventually release?**

  Decisions about names have not been made.

- **What will happen to the MS-DOS product line?**

  Microsoft will continue to enhance MS-DOS as long as customers require it. Future versions will be derived from the protected-mode technology developed in the Chicago project. Current MS-DOS–based applications and drivers will continue to be compatible with new versions of MS-DOS.

Architecture

- **Your performance goals sound very ambitious, considering all the functionality you're adding to Chicago. How will you achieve those goals?**

  Chicago will implement new technologies to better manage working set components, optimizing the use of memory on low-end system configurations. The networking, disk, and paging caches will be fully integrated to scale better as more memory is added to the system. Protected-mode device drivers will be dynamically loadable to ensure that only immediately-needed drivers are consuming memory. More components of the base operating system will be pageable. Great attention will be paid to effective memory page tuning, including hand-tuning source code.

- **I've heard Chicago described as a 32-bit operating system, yet I've also heard that portions of Chicago are implemented with 16-bit code. Are both these statements correct?**

  Chicago is a 32-bit preemptive multitasking operating system that implements some 16-bit code to provide compatibility with existing applications. Chicago's design deploys 32-bit code wherever it significantly improves performance without sacrificing application compatibility. It also retains existing 16-bit code where it is required to maintain compatibility or where 32-bit code would increase memory requirements without significantly improving performance. All of Chicago's I/O subsystems and device drivers (such as networking and file systems) are fully 32-bit, as are all the memory management and scheduling components (including the kernel and virtual memory manager). Many functions provided by the Graphics Device Interface (GDI) have been moved to 32-bit code, including the spooler and printing subsystem, the rasterizer, and the drawing operations performed by the graphics "DIBengine." Much of the window management code (User) remains 16-bit to retain application compatibility.

User Interface

- **How will the new user interface make the PC easier to use?**

The goal for the user interface for future versions of Windows is to make computers easy to use for all people, in order to expand the PC market to new users and new kinds of uses. Providing ease of use means that inexperienced users will find the user interface easy to learn and experienced user will find it more efficient and customizable. Chicago's user interface design will achieve these goals through the most extensive usability testing effort ever (thousands of hours of laboratory testing with hundreds of users of all levels of experience) and through feedback from various sources, including testing at customer sites, reviews with Windows training experts, audits by user interface consultants, feedback from focus groups, and analysis of product support calls.

Inexperienced PC users face a number of obstacles to using their PC today. Procedures which more experienced users take for granted, such as double-clicking with the mouse, are not intuitive. Managing multiple windows on a screen can be confusing, especially when some windows may completely cover others, or when clicking on a specific part of the window can cause it to apparently disappear, even though it is still present as a "minimized" window. Organization of files into hierarchies can make it difficult for inexperienced users to find information, and the restrictions of the 8.3 naming convention limit the amount of descriptive information that can be attached to a filename. Many people don't take full advantage of the capability to task-switch between different applications because the procedure for task-switching is not easy to discover.

Experienced PC users may find procedures for manipulating information and system resources cumbersome under existing products. Using different tools to work with different kinds of resources (such as files, programs, printers, and Control Panel settings) is inefficient. Integrating different kinds of information in a single document means that the user spends more time thinking about how to use different applications and less time on the content of the document itself. The ability to customize the system configuration is too limited for the most experienced users.

Both inexperienced and experienced users will find that the changes being made to the Windows interface in Chicago make Windows even easier to learn and use. The system "Taskbar" will make all the functions that most users ever need accessible with a single click of a button. The Taskbar will show all open windows, and make it much easier to switch between windows by just clicking on a button representing that window. Instead of mastering different kinds of tools (Program Manager, File Manager, Print Manager, Control Panel) to work with different resources on their computers, users of Chicago will be able to browse for and access all resources in a consistent fashion with a single tool. All resources in the system will have "property sheets" which present a tabbed notebook-style interface settings that can be directly changed. The implementation of OLE 2.0 in the system—which provides direct drag-and-drop manipulation of data throughout the user interface—will make working with documents easier and more intuitive.

- **Won’t a new user interface mean a lot of retraining for current Windows users?**

  The Chicago user interface is being designed to make people who are comfortable with the Windows 3.*x* interface immediately productive. Chicago’s smart setup technology will use the current system settings to present an initial configuration that is familiar for the current Windows user. For corporate customers and individuals who want to move more gradually to the new user interface, Chicago will enable them to continue running Program Manager and File Manager while they become more familiar with the new user interface features.

Plug and Play

- **What is Plug and Play? What benefits does Plug and Play provide?**

  Plug and Play is a technology jointly developed by PC product vendors that will dramatically improve the integration of PC hardware and software. Plug and Play is a general framework that advances the state of the PC architecture by defining how the software communicates with any device connected to the PC.

  With a Plug and Play PC, a user can easily install or connect Plug and Play devices to the system, letting the system automatically allocate hardware resources with no user intervention. For example, by simply plugging in a CD-ROM and sound card, a desktop PC can be easily turned into a multimedia playback system. The user simply plugs in the components, turns on the PC, and “plays” a video clip.

  Plug and Play also enables new system designs that can be dynamically reconfigured. For example, imagine a docking station that enables you to remove the portable system while it is still running so that you can take it to a meeting, and the system automatically reconfigures to work with a lower-resolution display and adjusts for the absence of the network card and large disk drive. Or imagine an infrared (IR)-enabled subnotebook that automatically recognizes, installs, and configures an IR-enabled printer when you walk into the printer room, so your applications are ready to print to that printer immediately.

  Plug and Play will also reduce costs for customers because it will save development and support costs for the product manufacturer. Today, as many as 50 percent of support calls received by operating system and device manufacturers are related to installation and configuration of devices. In addition, device driver development is simplified with Plug and Play because device manufacturers can write one driver that works across multiple bus types using the universal driver model specified by the Plug and Play architecture. Today, device manufacturers have to include bus-specific code in each of their drivers. With Plug and Play, specific bus configuration data is contained in “bus drivers.” Also, operating system preinstallation and configuration are simplified for OEMs because Plug and Play devices will automatically install and configure during Setup.

- **Will Plug and Play devices work with my current system or will I need a new system? What benefits will I receive when I purchase a Plug and Play device with my current system after I have installed Chicago on my system?**

  Plug and Play devices will provide complete backward compatibility to work with systems that were not designed according to the Plug and Play specification and may not be running a Plug and Play operating system like Chicago.

When you purchase a Plug and Play device for a legacy system running Chicago, you will be able to benefit from the automatic installation features of Plug and Play add-on devices. To take advantage of the dynamic configuration features of Plug and Play (as described above in the example about Plug and Play docking systems), you must purchase a system which has a Plug and Play BIOS as well as Plug and Play devices and operating system.

Win32 API Support

- **I've heard that Chicago implements a 32-bit API. Is that API different from the 32-bit API implemented on Windows NT?**

  There is only one 32-bit Windows API, called Win32, with different levels of functionality implemented on Windows 3.1, Chicago, and Windows NT. Chicago implements a large subset of the functionality of the Win32 API offered on Windows NT, and extends the Win32 API in some areas. These extensions will be delivered on Windows NT shortly after the release of Chicago.

- **If there are different implementations of the Win32 API available on different platforms in the Microsoft operating system product line, does that mean software developers will need different versions of their applications for Windows and Windows NT?**

  No. By following some simple guidelines, software developers can develop a single executable file that runs on Windows 3.*x*, Chicago, and Windows NT. At the recent Win32 Professional Developers' Conference, 32-bit applications from third-party developers were demonstrated running across the entire Windows family. In addition, Microsoft has provided in-depth technical sessions to explain the proper way to design applications for all 32-bit Windows platforms and supplied tools in the *Win32 SDK* to help make such development easier.

Availability of Applications for Chicago

- **When will applications that exploit Chicago be available? Won't that take a long time?**

  Software developers who are developing 32-bit applications for Windows 3.1 and Windows NT using the guidelines provided by Microsoft will have applications that are able to run on Chicago as soon as Chicago ships. There are already hundreds of 32-bit Windows applications available, with more being released every day. Many of the leading industry software vendors have already begun development of 32-bit applications for Chicago. Other ISVs will wait until Chicago ships to release 32-bit applications. Usually, those applications finish final testing and ship within 90 days of the operating system ship date.

Networking

- **Will I need new networking software to connect Chicago to my network server?**

  No. Chicago will continue to run existing real-mode networking components while enhancing the 32-bit protected-mode networking components first delivered with Windows for Workgroups.

- **What improvements will Chicago's networking support offer over Windows for Workgroups 3.11?**

  Chicago will enhance the open, flexible, high-performance 32-bit networking architecture offered today with Windows for Workgroups 3.11 that enables customers to mix and match networking components. Chicago will support NDIS 2.*x*, 3.*x* and ODI drivers, and will provide 32-bit NetBEUI, IPX/SPX, and TCP/IP protocols. Redirectors for SMB and NCP-based networks will be included. In addition, Chicago's new multiple-provider interface will make it possible for the user to view, browse, and connect to multiple networks in a consistent fashion.

- **Are you working with Novell on NetWare® support?**

  Microsoft is developing a 32-bit NCP Redirector that is seamlessly integrated with the Chicago user interface, and is encouraging Novell to do the same. Microsoft has offered Novell access to information and assistance to write a Chicago redirector. Novell engineers attended the Win32 Professional Developers' Conference and have been provided access to the *Preliminary Developer's Kit* for Chicago.

  With this approach, customers should be able to choose from multiple sources for reliable, high-performance NetWare client software when Chicago is released.

- **Will there be a Chicago server?**

  Chicago will not provide a separate "server" product such as Windows NT Advanced Server. For most server applications, and in the sense that most people ask about a "server product," Windows NT Advanced Server is the Microsoft server product.

  Chicago will continue to improve upon the peer server capabilities offered in Windows for Workgroups by offering additional features for remote installation, control, and administration. These features will make Chicago an even better product for an easy-to-use file- and print-sharing LAN that is ideally suited for a small-business, a small-department, or a remote office.

- **Can Chicago serve as an Internet client?**

  Yes. All the networking support you need to connect to the Internet is built into Chicago. Chicago includes a fast, robust 32-bit TCP/IP stack (TCP/IP is the language used by the Internet) and PPP, or "dial-in" support. Chicago is Internet-ready, whether you dial into a commercial Internet provider or you have access to the Internet via your corporate network. In addition, Chicago supports the large number of public domain tools used to connect to the Internet—such as Mosaic, WinWAIS and WinGopher—through the Windows Sockets programming interface. Chicago also includes utilities to help customers take advantage of the Internet, such as telnet and ftp.

Systems Management

- **What changes are you making in Chicago to make the operating system more manageable?**

  A primary goal for the Chicago project is to make Windows less expensive to deploy in an organization. Numerous studies have shown that the majority of the costs of owning a PC are in the costs of managing the PC. This includes installation, configuration, and maintenance of hardware and software, and MIS support for users. Chicago will provide an architecture for the delivery of tools that will make it easier for system administrators to install, configure, monitor, maintain, and troubleshoot their Windows-based desktops.

The foundation for a desktop management infrastructure is a store of data about the resources which require management. The Windows System Registry will provide a single database for information about the system, its hardware devices, its software applications, and all user-specific settings. Desktop management applications will be able to retrieve the information they require through an open interface of Registry API's. Access to this information will be protected with a "pass-through " implementation of a user-level security model, thus making management easier for the system administrator. Resource information will be accessible over the network using industry-standard Remote Procedure Call (RPC) technology.

- **What specific desktop management features will Chicago enable?**

Implementation of the desktop management architecture (described above) will enable Microsoft and third-party developers to deliver the functionality needed for managing the desktop. This includes support for hardware inventory monitoring, system software distribution, desktop and user configuration management, desktop backup and restoration, application software distribution, application use metering and licensing, and network and user problem diagnosis.

Hardware installation and configuration will be made much easier and less costly with the implementation of the Plug and Play architecture in devices and systems. The Windows Registry will provide data about hardware resources which can be accessed by third-party vendors to provide inventory management solutions.

The Chicago operating system itself can be set up from a network server and can be configured at the desktop to run locally or across the network. In each case, the administrator can establish a specific configuration for the installation, controlling which features are installed and which features can be accessed or altered by the end-user. Chicago will require only a floppy drive to start up, and paging of components to a swapfile on the network can be disabled to minimize network traffic.

Once Chicago is installed, administrators will be able to centrally configure desktop settings such as file and printer sharing, network access, and passwords. By running peer services, administrators can remotely monitor Chicago desktops to determine what resources are shared, what connections have been made, and what files are being used. Chicago enhances the security provided by Windows for Workgroups to include user-level security. To enable users to access their personal groups, applications, and data from any system on the network, Chicago will provide user profiles. These profiles will be centrally stored, accessed when the user logs in to a Chicago system, and used to install the appropriate configuration for that user.

Application software distribution solutions will be facilitated with the inclusion of a backup agent and peer server at each Windows desktop which can be used to download files from a server to the desktop. Future versions of Windows will provide support for the Licensing Services API to enable more effective application usage metering and licensing solutions for the desktop.

Chicago will provide a network diagnostic utility that will graphically display network traffic information for diagnosis by the network administrator. This information can be exported to products which provide additional network troubleshooting analysis. End-user problems will be easier to diagnose with a tool for remotely viewing the contents of the system Registry to identify installed hardware and software settings. Chicago also provides a "fail-safe boot" capability which will get a minimal system configuration up and running so a support professional can use the above configuration tools to analyze problems.

Mobile Computing

- **What improvements will Chicago offer for people who use a mobile or remote computer?**

Chicago will provide great support for mobile form-factor devices and will make it easy for end-users to access the resources on their desktop systems when they are away from the office. The implementation of Plug and Play in Chicago will support insertion and removal of devices such as PCMCIA cards while the operating system is running. It will also support automatic reconfiguration of dockable computers when they are inserted or removed from the docking station, without rebooting the system. An enhanced version of Advanced Power Management (APM) will further extend battery life. The services provided by Windows for Pen Computing will be enhanced and incorporated into Chicago to include basic inking and rendering support.

A special focus will be on remote networking. Any Chicago PC will be able to serve as a Remote Access dial-up server or a remote client for Windows NT Advanced Server, Novell NetWare server, or Chicago peer server. The same technology will be used for serial cable and infrared connections between PCs. The Remote Access architecture will be integrated with the Chicago networking architecture by using the same network protocols and advanced security features. Remote Access will support wireless devices and allow application developers to make their applications “slow-link aware” to improve the user experience when working on a remote system via modem rather than on a high-bandwidth network. Furthermore, Chicago will provide a simple form of file synchronization and APIs for applications to access the file synchronization services to merge changes when both the source document and copy have been modified. Remote email and Microsoft at Work™ Fax capability will also be included.

- **How are Chicago’s remote client capabilities different from Windows for Workgroups?**

Windows for Workgroups can remotely connect point to point (that is, from client to server) only with a Windows NT server. Chicago can connect not only to a Windows NT Remote Access Server but also to NetWare servers running NetWare Connect, other Chicago PCs, and network devices like a Shiva Netmodem (by using the PPP support in Chicago Remote Access). Chicago’s Remote Access architecture is open and extensible to enable third parties to provide additional capabilities, such as enhanced security modules.

For More Information

- **Where can I get the latest information on Chicago, directly from Microsoft?**

Microsoft has established a number of easily accessible electronic distribution points for new whitepapers, press releases and other pertinent documentation. Use the following electronic addresses to access further information:

On the Internet	*ftp.microsoft.com/peropsys/win_news*
On the Worldwide Web	*http://www.microsoft.com*
On Compuserve	*GO WINNEWS*

Index