IBM User's Guide to Wabi

Document Number GG24-4304-00

June 1994

International Technical Support Organization
Austin Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xiii.

First Edition (June 1994)

This edition applies to Version 1.1 of Wabi which is Feature Number 0146 of the AIXwindows Product Number 5601-257 for use with the AIX Version 3 Operating System.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 948S Building 821 Internal Zip 2834
11400 Burnet Road
Austin, Texas 78758-3493

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This document supplements the information available in the Wabi 1.1 product documentation. The information contained in this guide includes:

- How to install, configure and use Wabi 1.1 on IBM systems.
- A high level explanation on how Wabi 1.1 works.
- Which Windows applications are supported by Wabi 1.1
- Sample performance results for Wabi 1.1 running on some IBM systems.
- Answers to frequently asked questions and common problems.

This document was written for customers and system engineers who need to know how to set up and use Wabi 1.1 on IBM systems. A basic knowledge of AIX, DOS and the Microsoft Windows environment is assumed.

AX (132 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help customers, industry specialists, systems engineers set up and use Wabi 1.1 on IBM systems. The information in this publication is not intended as the specification of any programming interfaces that are provided by Wabi 1.1, AIX Version 3.2.5, AIXwindows Environment 6000 Version 1.2.5 or AIX Personal Computer Simulator/6000 Version 1.2. See the PUBLICATIONS section of the IBM Programming Announcements of Wabi 1.1, AIX Version 3.2.5, AIXwindows Environment 6000 Version 1.2.5 and AIX Personal Computer Simulator/6000 Version 1.2 for information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

Any performance data contained in this document was determined in a controlled environment, and therefore, the results that may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data for their specific environment.

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to

the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
AIXwindows	IBM
PowerPC	RISC System/6000
Xstation Manager	

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

1-2-3, AmiPro, Lotus	Lotus Development Corporation.
AFS	Transarc Corporation.
AfterDark	Berkley Systems
Aldus PageMaker	Aldus Corporation
CorelDRAW	Corel Corporation
FrameMaker	Frame Technology, Inc.
Harvard Graphics	Software Publishing Corporation
Microsoft, Microsoft Excel, Microsoft PowerPoint, Microsoft Project, Microsoft QuickWin, Microsoft Windows, Microsoft Word	Microsoft Corporation
Motif, OSF	Open Software Foundation
Network File System, NFS, Sun, Sun Microsystems, SunOS, SunSelect, Wabi	Sun Microsystems, Inc.
Novell, Netware	Novell, Inc.
Paradox, Quattro Pro	Borland International
PROCOMM PLUS	Datastorm Technologies, Inc.
PostScript, Adobe Type Manager	Adobe Systems, Inc.
UNIX	UNIX System Laboratories Inc.
WinTach	Texas Instruments Corporation.
WordPerfect	WordPerfect Corporation
X Window System	X Consortium

Preface

This document explains how to set up and use Wabi 1.1 on IBM systems.

It contains explanations on how to install, configure and use Wabi 1.1. Also included is a high-level explanation on how Wabi 1.1 works, information on which Windows applications are supported, and sample performance results for Wabi 1.1 running on some IBM systems.

This document is intended for customers and system engineers who need to know how to set up and use Wabi 1.1 on IBM systems.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Introduction”

  This includes a description of Wabi, and why you would use it.

- Chapter 2, “How Wabi Works”

  This chapter describes how Wabi allows Windows software to run unmodified on a RISC System/6000 and discusses instruction emulation, native library implementation and filename translation.

- Chapter 3, “Installation and Setup”

  This provides a guide intended to help experienced AIX users get Wabi installed and running quickly. We then give more detailed instructions on installing Wabi, Microsoft Windows and Windows Applications.

- Chapter 4, “Application Support”

  This chapter provides information on application support under Wabi. It lists the applications that are qualified as working correctly with Wabi, and lists our experiences using some unqualified applications.

- Chapter 5, “DOS Emulation”

  This describes the requirements for interfacing Wabi with AIX Personal Computer Simulator/6000.

- Chapter 6, “Devices”

  This chapter details the use of printer, disk, network and other devices under Wabi.

- Chapter 7, “X Windows Support”

  This chapter explains how Wabi is supported on a graphical display under the X Window system, including the use of fonts, colors and color maps and the behavior of Wabi under X Windows window managers. We also discuss methods of creating desktop icons or Motif Window Manager icons for starting Windows applications.

- Chapter 8, “Using Wabi in a Multiuser Environment”

This chapter provides guidelines on running Wabi in a multiuser environment including reducing repetitive installations, sharing applications and data, and controlling access to applications on a group-by-group basis.

- Chapter 9, “wabi.ini File”

  This chapter explains the contents of the Wabi initialization file and how to modify this file.

- Chapter 10, “Performance”

  In this chapter we give some guidelines on the relative performance of Wabi on various RISC System/6000 models and graphics adapters. We will also give some memory usage results that indicate how Wabi's performance is effected by available system memory.

- Chapter 11, “Wabi Around the World”

  This chapter provides information on setting up Wabi in languages other than English. It also demonstrates a method of selecting applications to run by looking at the LANG environment variable for users that operate in multiple languages.

- Chapter 12, “Frequently Asked Questions”

  This chapter answers many of the questions most commonly asked by Wabi users.

- Chapter 13, “Common Problems”

  This chapter provides solutions to some of the most common problems that are encountered by Wabi users.

- Appendix A, “Wabi File Layout”

  This appendix lists the files that make up Wabi, and the functions or contents of each.

- Appendix B, “Wabi 1.1 Manual Page”

  This appendix is a copy of the Wabi 1.1 manual page that is delivered with Wabi.

- Appendix C, “Redbook Sample Tools and Utility Programs”

  This appendix details several utility programs and shell scripts that we wrote as part of this residency.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *Wabi 1.1 for AIX User’s Guide*, SC23-2643
- *IBM AIX Version 3.2 for RISC System/6000 General Concepts and Procedures*, GC23-2202
- *AIX Version 3.2 for RISC System/6000 Installation Guide*, SC23-2341
- *AIX PC Simulator/6000 Guide and Reference*, SC23-2452

International Technical Support Organization Publications

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

Acknowledgments

The advisor for this project was:

Mark Kressin
International Technical Support Organization, Austin Center

The authors of this document are:

Catherine Chevallier
IBM Paris, France

Cameron Ferstat
IBM Perth, Western Australia

This publication is the result of a residency conducted at the International Technical Support Organization, Austin Center.

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Paul Lugo
Brian Pohl
Emulator Product Test Team
Power Personal Systems

Barbara Lennan and Janice Hawley
Huy Nguyen
International Technical Support Organization, Austin Center

Nina Vogl
Worldwide AIX Technology Transfer Group, Austin Texas

The participants on the IBM Wabi forum

Chapter 1. Introduction

Wabi** is an environment that allows off-the-shelf or shrink-wrapped Microsoft Windows** 3.1 compliant applications to run on the AIX* operating systems. This allows users to choose from the many excellent Windows productivity applications, such as spreadsheets and word processing programs, while maintaining the power and networking flexibility inherent in today's AIX workstations.

Wabi users can run their existing Windows Applications concurrently, without modification, on the same workstation or X terminal they use to run their AIX business and technical applications.

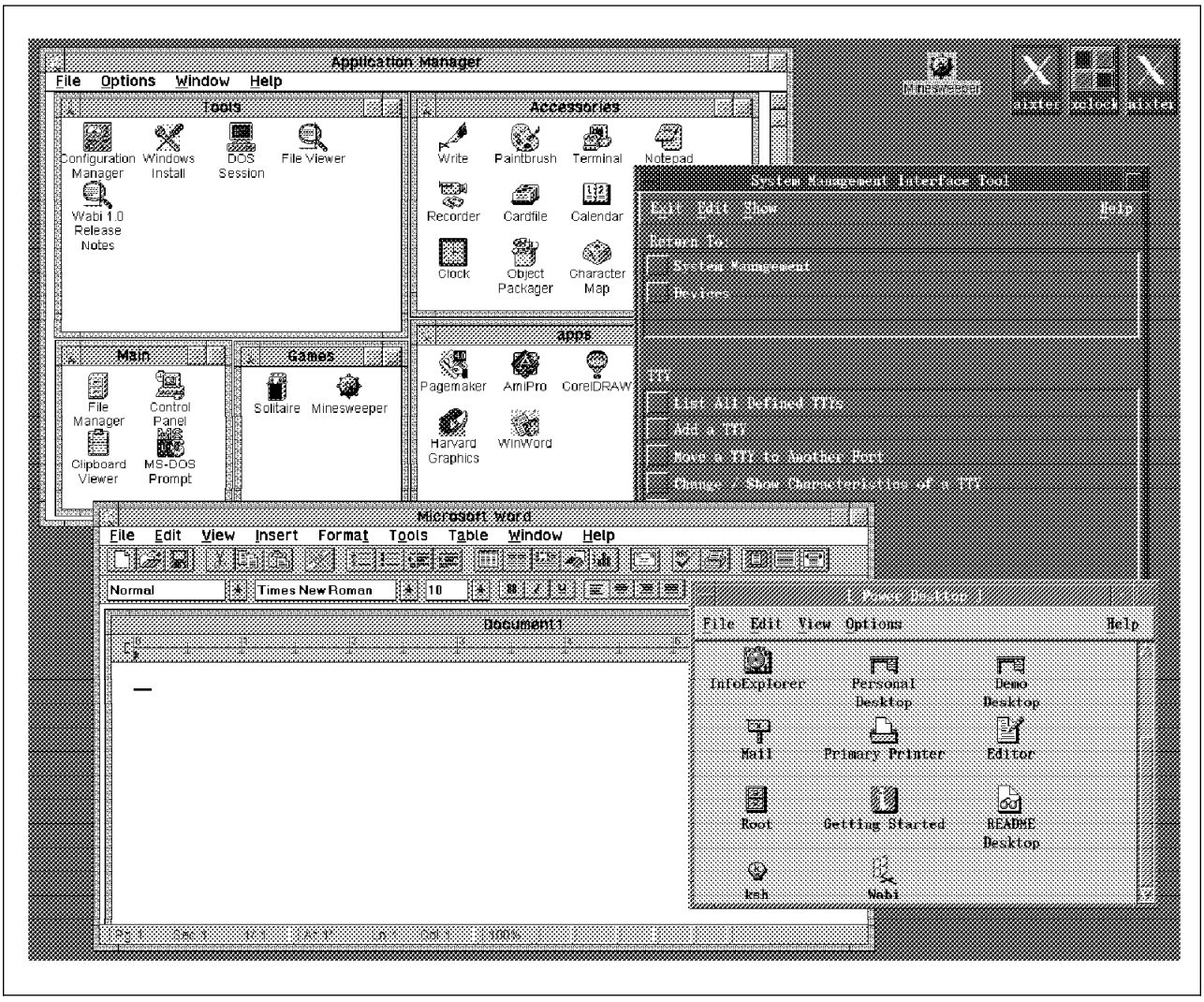

Figure 1. Typical Wabi Environment

Figure 1 shows a typical Wabi environment. On a single workstation screen we can see the Wabi Application Manager with several application groups, a Windows program (Microsoft Word**), and an AIX application (System Management Interface Tool). These are all running under IBM* AIXwindows* Environment/6000 and are shown here with the AIXwindows Desktop, which includes an icon for launching Wabi. Windows applications can also be launched directly by clicking on desktop icons.

The initial release of Wabi is Version 1.1. Wabi 1.1 includes:

- An 80386 instruction emulator to execute the Windows binary programs.
- The Microsoft Windows 3.1 Application Program Interface (Win16 API) re-implemented in native UNIX** code.

  The most critical Windows Dynamically Linked Libraries (DLLs) have been re-implemented under AIX and compiled as IBM RISC System/6000* binary programs.
- A shell program named Application Manager similar to the Windows Program Manager shell.

  The Application Manager allows you to run programs and create application groups and icons.
- A set of tools for configuring Wabi, installing Microsoft Windows 3.1 code, viewing files and initiating a DOS session (although the DOS emulator is not supplied).

1.1 Why Use Wabi?

Some benefits of using Wabi include:

- An increased range of applications

  Choose from the large range of Microsoft Windows 3.1 compliant applications, and over 6,500 AIX/6000* applications on a single workstation screen. This single screen could even be a laptop system.
- The ability to cut and paste between UNIX and Windows applications
- No need to have both PC and UNIX workstations on your desk
- Impressive Windows performance that can exceed many popular PC models
- A UNIX server with X-Terminals running Wabi can often be more cost effective and easier to manage than separate PCs on each desk
- Access to both local and remote UNIX filesystems
- Access to and integration with AIX devices such as printers, serial ports and CD-ROM drives
- A phased migration from PC architectures up to high performance RISC System/6000 workstations allowing users to preserve some or all of their Windows applications in the short or long term
- Other benefits common to network server systems such as ease of backups and security

1.2 What Does Wabi Stand For?

Ever since its initial announcement, there has been continual confusion over the name of Wabi. Many people believe that Wabi is an acronym, standing for (among other suggestions):

- Windows Application Binary Interface
- What A Brilliant Idea
- What About Bad Interfaces

In fact, Wabi is not an acronym at all. Wabi is a trademarked name only. The Wabi trademark is owned by SunSelect** (a Sun** Microsystems Inc. business), and should be written with a capital *W* and small *abi*.

Chapter 2. How Wabi Works

This chapter explains how Wabi operates. 2.1, “Overview” is intended as a fairly high level overview explanation while a more detailed explanation on Wabi operation is presented in 2.2, “More Detail.” The following sections explain more specific areas of Wabi operation.

2.1 Overview

Wabi software is called *middle-ware*. It provides interfaces between the unmodified Windows binary program and the native operating system (here, AIX/6000). This is shown in Figure 2. Wabi takes service requests from Windows applications, that would normally be handled by DOS and Windows and redirects them to similar UNIX and X Window System** calls for processing. For example, if a Windows application program issues a request for data to be retrieved from a disk file, Wabi takes that request and redirects it to the appropriate AIX service for retrieving disk data. The data is passed to Wabi from the AIX service which then repackages the data into the Microsoft Windows 3.1 format expected by the application and presents the data to the application. This redirection process is completely transparent to the Windows application - all the application knows is that it issued a standard Windows file request and received the data back in the standard Windows format.

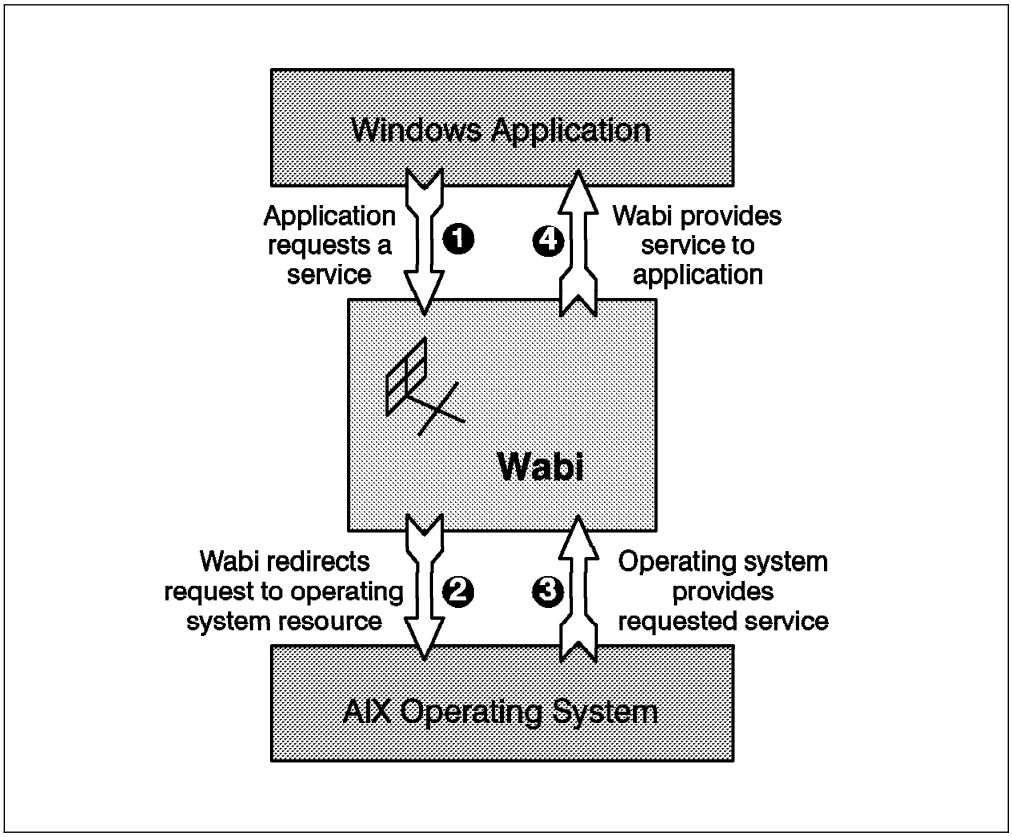

Figure 2. How Wabi Operates as Middle-Ware

An easy way to think about the way Wabi interfaces between the Windows application and the AIX operating system is to think of Wabi as a language translator - translating between French and English for example.

2.2 More Detail

Figure 3 illustrates the difference between running an application in the native Windows environment and running the same application under Wabi.

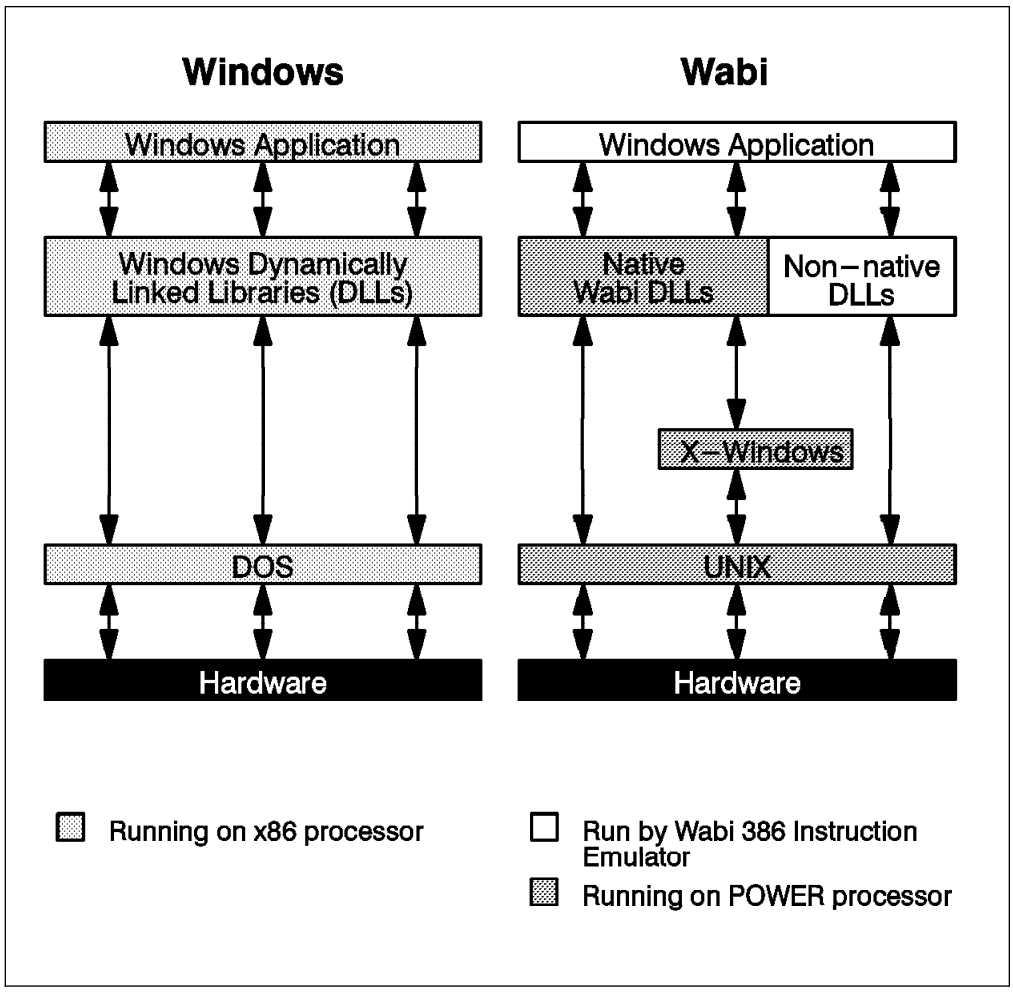

Figure 3. Service Calls in a Wabi vs Windows Environment

In the Windows environment on a PC using an x86 processor (for example a 386 or 486), the processor executes the instructions of the application program. Some of these instructions may include a call to one of the Dynamically Linked Libraries (DLLs) that are supplied as part of Microsoft Windows 3.1. The instructions that make up the DLLs are also executed by the x86 processor. When system resources such as disk access or input/output operations (I/O) are required, the Windows DLLs will make calls to DOS, which interfaces with the hardware.

In the Wabi environment on a UNIX platform, most of the important DLLs that are supplied as part of Microsoft Windows 3.1 have been re-implemented as native AIX programs. Thus, when the application program makes a call to one of these DLLs, Wabi translates the Windows library call into a native AIX library call. The called routine executes in native Power or PowerPC** code, achieving excellent

performance characteristics. Since up to 80 percent of the execution time of most Windows programs is spent executing routines in the DLLs, Wabi is able to achieve good overall application performance.

Some Windows DLLs have not yet been re-implemented in native code in Wabi 1.1. One example of this is the Dynamic Data Exchange DLL (DDEML.DLL) which allows live data to be shared between different Windows applications. These non-native DLLs will be provided when you install Windows or as part of particular applications. Applications that use these DLLs will still operate but since the DLLs are not re-implemented in native code, calls to routines in these DLLs will be executed by the 386 instruction emulator and thus will not perform as quickly as the native DLLs.

The DLLs that have been implemented natively in Wabi 1.1 are:

GDI.DLL	Graphics Device Interface
KERNEL.DLL	System services such as multitasking, memory management and resource management
LZEXPAND.DLL	Data decompression library for decompressing files previously compressed by the file compression utility COMPRESS.EXE
USER.DLL	Window management - both for the overall Windows environment and for application windows
WIN87EM.DLL	Floating point emulation library
WINMEM32.DLL	32-bit memory addressing and management

In addition, the following DLLs do not have native implementations but can operate under the 386 instruction emulator:

COMMDLG.DLL	Procedures and templates for common dialog boxes
DDEML.DLL	Dynamic data exchange (for data sharing/exchange between applications)
OLECLI.DLL, OLESRV.DLL	Object Linking and Embedding (also for data sharing between applications)
SHELL.DLL	Library which supports a registration database, drag-drop features, application associations and extracting icons from executable files
VER.DLL	Library for file installation and for analyzing currently installed files.

On occasions where Microsoft Windows 3.1 would make a call to DOS for system resources, Wabi makes a call to the underlying UNIX operating system which performs all the device access and provides system services.

All graphical operations are implemented as X Window calls. This allows Wabi to operate in a distributed X Windows environment and to operate to some degree with common X Windows window managers such as the Open Software Foundation's OSF** Motif** window manager, and desktop environments such as the AIXwindows Desktop. For more detail on Wabi's operation under the X Window System, see Chapter 7, “X Windows Support” on page 51.

2.3 Windows Application Program Interface

There are several versions of the Windows Application Program Interface (WinAPI). These are shown in Figure 4.

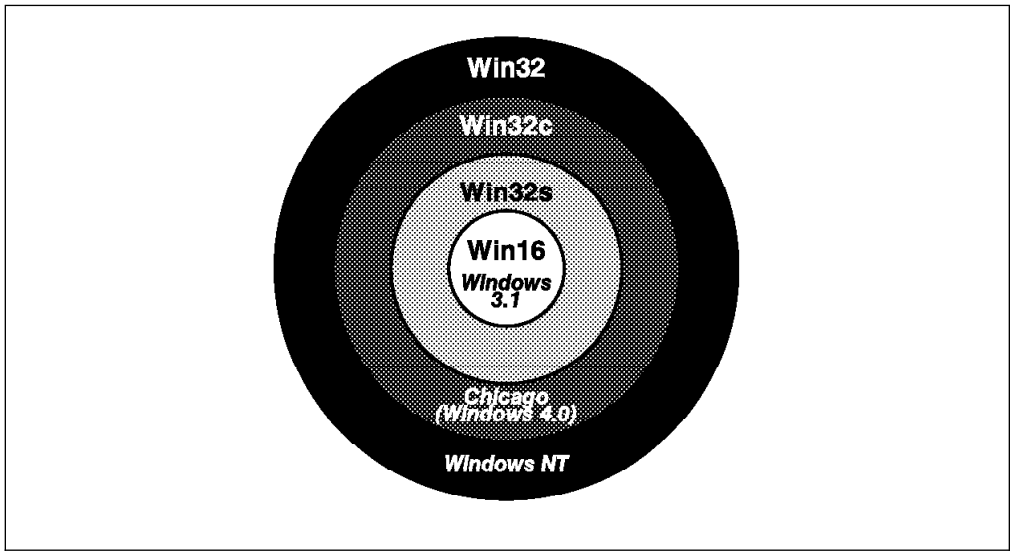

Figure 4. Windows Application Program Interface Versions

Win16 This is the 16-bit Windows API as implemented in Microsoft Windows 3.1 The great majority of current Windows applications are written to to this interface specification.

Win32 This is the 32-bit interface implemented in the Windows NT operating system. Allowing 32-bit operations achieves large performance benefits, particularly in numerical or mathematical operations. At present, Windows NT is being marketed as a fully featured operating system with preemptive multitasking, threads, an enhanced filesystem and multiprocessor support, however because of the high resource requirements, NT is mostly limited to server environments and is not seen as a replacement for Microsoft Windows 3.1. There are relatively few applications that are currently written to use the Win32 specification.

Win32s This is a set of Dynamically Linked Libraries (DLLs) intended to allow 32-bit applications written to take advantage of the benefits of 32-bit operations to operate (though with lower performance) on the 16-bit Microsoft Windows 3.1 operating system. This is achieved by *thunking* the 32-bit calls to 16-bit calls. The thunking process involves copying the 32-bit stack to a 16-bit stack and converting 32-bit memory pointers to 16-bit. This process introduces a performance overhead for applications that make many calls to the Windows DLLs.

The *s* in Win32s stands for *subset*. This indicates that the Win32s API implements only some of the calls in Win32. Win32s does not provide support for threads, security and multiprocessing.

Win32c This is the name for the API that will be implemented in the future Microsoft "Chicago" operating system (also referred to as Windows 4.0). This is intended to be Microsoft's replacement for Windows 3.1 as a desktop or workstation operating system. It will have lower resource requirements than NT.

The Win32c API is again a subset of the full Win32 interface. It is intended to support threads, but not multiprocessing or security.

Wabi 1.1 only supports the Win16 Application Programming Interface. This is the interface used by most of today's Windows applications.

2.4 Memory

All Windows applications running under a single instance of Wabi are run as if they are running on the same PC. This means that the applications share a single memory address space. This implementation was chosen because many Windows applications use shared memory as a method of interprocess communication. This is only possible if they share the same address space. The downside of this implementation is that if a Windows application crashes, you will probably have to stop and restart your entire Wabi session. This is not uncommon in a real Windows environment.

Since Microsoft Windows 3.1 runs in a DOS environment, memory management is complicated by the limitations inherent in DOS. Memory is divided into three different types:

2.4.1 Conventional Memory

Conventional memory is the original, basic memory that has always existed since the birth of the PC. The original DOS versions only new about conventional memory, which at that time was probably around 256 Kilobytes (KB). Because of the architecture of the first DOS systems, which were based on Intel 8086 processors, and the design of DOS, conventional memory is limited to 640 KB. In a regular DOS environment, some of this memory is used by DOS, and more is used by device drivers and Terminate and Stay Resident (TSR) programs. As DOS and PCs became more popular, people began running complex programs under DOS and with rapidly falling memory prices, memory requirements quickly rose. The 640K limit quickly became a real restriction in this environment.

2.4.2 Extended Memory

With the introduction of the 286 processor, the width of the Address Bus was increased to allow the chips to address greater amounts of memory - up to a total of 16 MB. However, due to the size of some internal registers within the 286 chip, a different operating mode had to be introduced to allow access to this *extended* memory. This new mode also added features to protect memory from being modified by faulty programs, and thus was named protected mode. The original, basic mode of operation was referred to as *real* mode. There was very little support added in DOS for accessing memory through protected mode and thus it was necessary to use an extended memory manager program to facilitate and control the access to extended memory.

Microsoft Windows 3.1 requires at least 1 MB of extended memory to operate. The HIMEM extended memory manager is included with Windows and allows you to load most of Windows and its device drivers into extended memory, leaving most of the 640K conventional memory available for any DOS applications running under Windows. To Windows applications running on a system there is effectively no distinction between conventional and extended memory - they appear to the application as a single contiguous block of memory.

2.4.3 Expanded Memory

Expanded memory exists separately from a system's conventional and extended memory. It is generally installed on an expanded memory board and requires an expanded memory manager to be used. The expanded memory manager maps the expanded memory onto the upper memory area from 640KB to 1MB which is normally used by the DOS operating system. Applications request access to expanded memory from the expanded memory manager. The application must be specifically written to interact with an expanded memory manager if it is to use expanded memory. Using expanded memory is generally slower than using extended memory.

Windows and Windows applications don't use expanded memory, however DOS applications that have been specifically written to use expanded memory can be run under Windows and may still access expanded memory if it is available. On 386 and 486 based PCs, *expanded memory emulators* can use extended memory to simulate expanded memory for applications that require it. Windows running in enhanced mode (see 2.5, “Windows Modes”) can use the EMM386 expanded memory manager to simulate expanded memory.

2.4.4 Wabi's Memory Model

This memory model is greatly simplified under Wabi. Since Wabi and the various RISC System/6000 architectures do not suffer from the limitations of DOS and the 8086, 8088 and 80286 processors, there is no distinction required between conventional and extended memory. The Windows applications simply request memory in the same way as they would running under Windows, and Wabi, in turn, requests this memory from AIX.

The memory that is available to Windows and Windows applications running under Wabi will be limited by the following factors:

- The size of the system memory and paging space on your RISC System/6000.
- The other applications running on your system.
- The per user memory limits that can be set under SMIT.

Since expanded memory is not used by Windows and Windows applications, there is no requirement for expanded memory support under Wabi. It is possible to use DOS applications that require expanded memory under the AIX Personal Computer Simulator/6000. For more information on using expanded memory under AIX Personal Computer Simulator/6000, see the PC Simulator/6000 Guide and Reference.

2.5 Windows Modes

There are three different modes of operation that are defined for Microsoft Windows. These modes are explained below:

Real Mode

Real mode operation is not supported on Microsoft Windows 3.1 but was used up until Windows 3.0. Real mode simply means that the CPU is kept in real mode, disabling access to extended memory. Windows 3.0 was able to operate in this manner to provide compatibility with hardware containing only 640KB of conventional memory. Task swapping in this mode was achieved by copying memory to disk. Although

functional, this process proved to be very slow on the older hardware. The additional memory requirements of Windows 3.1 and the performance demands of users meant that this mode was removed from Microsoft Windows 3.1

Standard Mode
: Standard mode is the default operation mode for Windows running on 286 processors. This allows Windows and Windows applications direct access to conventional and extended memory as one contiguous block. It is possible to run Microsoft Windows 3.1 in standard mode on 386 and better processors, however doing so denies the user the benefits of enhanced mode.

Enhanced Mode
: Enhanced mode operation, as well as providing support for extended memory, introduces the concept of virtual memory by swapping 4KB pages of memory to a file on the hard disk in a similar way to paging in a UNIX operating system. In enhanced mode, Windows uses an extended memory manager such as HIMEM.SYS to load itself and its device drivers into extended memory. Enhanced mode also provides true multitasking support, rather than the task switching that is provided with real and standard modes. This mode is only available on 386 or better processors as it requires special paging support on the processor.

Wabi does not support real or standard modes of operation, but operates only in enhanced mode. This is by far the most functional mode of Windows and is the mode most often used by Windows users on 386 or better PCs.

2.6 Reasons for Application Failure

Having read how Wabi natively implements several DLLs and runs most others, and how the Windows binary code in the Windows applications are executed by the Wabi instruction emulator, you may now be wondering why all Windows applications will not run under Wabi.

Four main reasons for applications failing under Wabi are:

- As described in 2.3, “Windows Application Program Interface” on page 8, Wabi 1.1 implements the Win16 Application Program Interface. If the Windows application is written to use one of the other levels of Windows APIs, Win32s, Win32c or Win32, it will fail when run under Wabi.
- As many experienced Windows users will know, there are many features of Microsoft Windows 3.1 that are not included in any official documentation. These may include features that were accidentally omitted from documentation, or were deliberately not documented because they did not agree with accepted or standard practices, or were not fully tested. The same applies to the Windows APIs. There are calls that exist in the API as implemented by Microsoft Windows 3.1 but do not appear in the formally published API documentation. Wabi fully implements the API as documented. Wabi also implements many of these undocumented calls and will implement more in future Wabi releases as they are discovered, to ensure that Wabi follows Microsoft Windows 3.1 behavior as closely as possible.

- Many Windows applications do not adhere fully to any of the Windows APIs. In some cases, an application was ported to Windows from DOS, and still maintains some native DOS or BIOS calls. Wabi 1.1 does not reside on top of DOS or use a PC BIOS, and calls to these will cause an application to fail.
- In some cases, usually for performance reasons, programmers will directly access a specific hardware device such as a video or sound adapter. As the underlying hardware in RISC System/6000 is radically different to that of a PC, these calls will fail.

2.7 Filename Translation

As most people know, DOS filenames are limited to an eight character name, and a three character extension. These are separated by a period (dot character). UNIX filenames are much more flexible than DOS filenames. If Wabi is to be able to share files with UNIX, there must be some method of mapping UNIX filenames to legal DOS filenames when they are being accessed under Wabi.

The method that has been agreed by SunSelect and IBM for use in Wabi 1.1 is based upon an X/Open standard for mapping filenames and attributes.

2.7.1 Legal DOS Filenames

A legal DOS filename will meet the following criteria:

- A filename of eight characters or less.
- An optional extension of three characters or less.

  The filename and extension are separated by a dot (.) character.
- All lowercase characters.

  Regardless of whether they appear as uppercase under DOS or lowercase under Windows, DOS filenames are stored within the DOS filesystem in lowercase. This means that they must appear in UNIX as fully lowercase. The exception to this rule is that DOS stores files on diskette in uppercase. It is possible with the AIX Personal Computer Simulator/6000 to specify whether the files should be stored in the AIX filesystem in uppercase or lowercase with the default being uppercase. When using PC Simulator in conjunction with Wabi, you should use the option to store the files in lowercase so they will be accessible from both PC Simulator and from Wabi.
- The characters , + [ ] * ? : \ / ; = < > are not permitted.

  A single dot character is permitted separating the filename and extension.

In addition, Wabi adds the following restriction:

- A DOS filename cannot include a tilde character (~) in the sixth character position.

  This restriction is due to the fact that Wabi uses a tilde character in the sixth position to signify internally that the filename must be translated back to its correct name.

2.7.2 Filename Translation

A UNIX filename that meets the requirements for a legal DOS filename will not be translated or mapped. This will hopefully be the most common case for files that are going to be used from within Windows applications.

Filenames that do not meet these requirements must be mapped to a legal DOS filename to allow the Windows applications to access these files. The mapping is performed using the following algorithm.

1. The first five characters of the filename are retained.

   If the filename is less than five characters it will be padded with tildes.

2. Uppercase characters are mapped to lowercase.
3. If there are any illegal characters in the first five positions, they are replaced by tildes.
4. A tilde (~) is added as the sixth character.
5. The seventh and eighth positions are used for two alpha-numeric mapping characters assigned by Wabi.

   The mapping characters are used to ensure that the mapped name is unique. For example, the UNIX filenames tweedledum and tweedledee share the same first five characters but must map to unique DOS names.

6. If the extension is a legal DOS extension, it is retained.

   If the extension is in any way illegal (for example, it contains uppercase or illegal characters) no extension will be used.

This mapping method ensures that the mapped name will always have a name exactly eight characters long. It may include a dot followed by a three character extension. The filename will always be unique within an instance of Wabi.

2.7.3 Examples

The following examples in Table 1 on page 14 show the operation of the Wabi filename translation method. In this table, the two mapping characters are represented by xx.

Table 1. UNIX to DOS Filename Translation Examples		
UNIX filename	**Mapped DOS filename**	**Notes**
abc123.def	abc123.def	No mapping required
a	a	No mapping required
A	a~~~~xx	Must be mapped as it is in uppercase
a_long_name	a_lon~xx	Keeps the first 5 characters
AB.c	ab~~~xx.c	Padded with tildes
Ab.c	ab~~~xx.c	The xx value will be different to the preceding value
abc.	abc~~xx	A filename with a dot (.) but no extension will be mapped
a.b.c	a~b~~xx.c	Assumes that the final dot is for the extension and replaces earlier dots with tildes
abcd.efgh	abcd~~xx	Illegal extension is not used
.login	~logi~xx	The hidden attribute is also set
cam~	cam~	No mapping required
cathy~	cathy~xx	Cannot have ~ in sixth position

2.7.4 Mapping Characters

Wabi generates two mapping characters which are used in the seventh and eighth positions of the mapped filename to ensure that the filename is unique, and to aid in mapping the filename back to its original name.

It is important to be aware that in Wabi 1.1, the current process identifier (PID) of the Wabi program is used in the generation of these characters. This means that if two different Wabi users look at the same file, which may be a file on a shared network drive, they will see different mapping characters and thus different filenames. For example, the UNIX file long_names_rule_OK may appear to one user as long_~e5 but may appear to the other user as long_~br. Also, if a user exits and restarts Wabi, the Wabi program will have a different PID and thus the mapping characters and the filename will be different.

Another problem with this method is there is no way of predicting what the mapping characters will be, and thus it can be very difficult to distinguish between two files with similarly named files. For example, the UNIX files similar_1 and similar_2 could be mapped under Wabi to the names simil~a7 and simil~e6. There is no way under Wabi to know which file has been assigned which name.

Finally, you can experience problems transferring files between hard disk and diskette. In most cases, Wabi will retain the original name of a file because it maintains a table of mapped names and will reverse-map any filename that has a tilde character in the sixth position. Thus you can use a Windows application to edit a file and save it back under the same name, or use the File Manager to copy a file from a hard disk to a network drive. In both cases, Wabi will retain the full original filename although you will only see the mapped name in the application or File Manager. However, if you copy the filename to a DOS diskette, the file is stored in a real DOS filesystem, and thus must adhere to legal DOS name

standards. Thus the filename is actually stored with its translated name. If you do not exit from Wabi, the name will still be in the mapping table and thus can copy the file back to a hard disk and retain the original name. However, if you exit and restart Wabi, the mapping table is lost and the filename cannot be remapped to its original name.

For these reasons, we strongly recommend that you restrict the names of files that will be used by Windows applications under Wabi to legal DOS names.

2.7.5 DOS File Attributes

When looking at a UNIX file under Wabi, the DOS file attributes are mapped or assigned as follows:

1. First the file permissions of the UNIX file are determined. Wabi decides if the user file permissions, the group file permissions or the other file permissions should apply.
2. If the user does not have write access to the UNIX file, the DOS read-only attribute is set.
3. If the UNIX file has the setuid bit set, the DOS hidden attribute is set.
4. If the UNIX file starts with a dot character (.), and the filename is not . or .. then the DOS hidden attribute is set.
5. If the UNIX directory attribute is set, the DOS directory attribute will be set.
6. The DOS archive attribute is set.

Note: The Properties dialog of the Windows File Manager is not supported in Wabi 1.1. The values in this dialog are not reflective of the actual file permissions or attributes. Also, a file's attributes cannot be modified using this dialog.

2.7.6 UNIX File Attributes

When a file is created under Wabi, the UNIX file attributes are set as follows:

1. The permission bits are first set to 666 (read and write for the user, group and others).
2. If the DOS read-only attribute is set, the UNIX write bit is cleared for user, group and others.
3. If the DOS hidden attribute is set, the UNIX setuid bit will be set.
4. The UNIX file permissions will then be set to either the permissions as defined above or the UNIX umask, whichever is more restrictive.

   For example, the standard AIX umask is 022 which means a file will be created with permissions read and write for the user, read for the group and read for others (-rw-r--r--). In this case, Wabi will not create a file with write permissions for the group and others. If the DOS read-only attribute is set, the permissions will be -r--r--r--. If the DOS read-only attribute is not set, the permissions will be -rw-r--r--.

Note: The Properties dialog of the Windows File Manager is not supported in Wabi 1.1. The values in this dialog are not reflective of the actual file permissions or attributes. Also, a file's attributes cannot be modified using this dialog.

Chapter 3. Installation and Setup

This chapter covers the installation of Wabi, Microsoft Windows 3.1 and Windows applications. The first section is intended as a guide for experienced AIX users to get Wabi installed and running quickly and easily. The installation is explained in more detail in the following sections.

More information on the installation and setup of Wabi can also be found in the Wabi 1.1 for AIX User's Guide.

A method is also given for copying installation files from diskette to a directory on disk. This method can save a lot of time if Windows or the same Windows applications have to be installed for many users.

For more information on installing and setting up Wabi on a multi-user system, see Chapter 8, “Using Wabi in a Multiuser Environment” on page 67.

3.1 Wabi Quickstart for Experienced AIX Users

Wabi is an easy product to install and begin using in a stand-alone configuration. The steps for installing and configuring Wabi are as follows:

1. Prerequisites
   a. AIX Version 3.2.5
   b. AIXwindows 1.2.5 (X11R5)
   c. If you have a GT3, GT3i, GT4e, GT4, GT4x, GT4i or Gt4xi graphics adapter, ensure that you have applied AIXwindows PTF U424296 to fix a problem with screen refreshing
2. Install Wabi
   a. Wabi is packaged as a standard AIX installation image. As the root user, install Wabi.obj using `SMIT` or the `installp` command. All other steps in the installation process should be performed as the Wabi user, and will need to be performed for each user. We do not recommend using root as a Wabi user.
   b. Add the directory /usr/lpp/Wabi/bin to the search path of your Wabi user.
   c. If you use the AIXwindows Desktop, run the following command after starting the desktop:

      ```
      $ /usr/lpp/Wabi/bin/xdt3wabi
      ```

      This will create a Wabi icon on your desktop for launching Wabi.
   d. From inside X Windows, start Wabi by typing the command:

      ```
      $ wabi
      ```

      at the command prompt, or by double clicking the Wabi icon.
   e. You should read the Wabi 1.1 Release Notes to obtain the most up to date information on this Wabi release. You can do this by double-clicking on the **Wabi 1.1 Release Notes** icon. The start of the release notes gives instructions on printing the notes.
3. Install Microsoft Windows 3.1

a. You may not need to install Microsoft Windows 3.1 to run your Windows applications. Of the 13 qualified Windows applications, only two applications require that Microsoft Windows 3.1 be installed:

   - Microsoft Powerpoint**
   - Paradox**

   The other reasons to install Microsoft Windows 3.1 are to obtain the applets and productivity aides that it includes, such as Write, Paintbrush, Terminal, Calculator, Notepad, File Manager and Games, or to obtain Windows Help functions.

b. To install Microsoft Windows 3.1, double click on the **Windows Install** icon in the tools group. You will be prompted for the Windows installation diskettes or for the directory or network drive containing images of the installation diskettes.

   Do not run setup from the A: drive to install Microsoft Windows 3.1. Because Wabi has natively implemented several of the Windows DLLs, it is necessary to use the Windows Install option which will copy the required files from the Windows installation media, and intersperse these with the appropriate files or links from the Wabi installation directories.

4. Install Your Windows Applications

   a. Windows applications are installed in the same way as with native Microsoft Windows 3.1 From the File pull-down menu, select Run... then click on **Browse...** Select the **[-a-]** drive in the Drives pull-down menu, and look at the executable files. In most cases, there will be either **SETUP.EXE** or **INSTALL.EXE**. Select this, click **OK**, then **OK** on the Run dialog box. For details on the installation options you should see the documentation for the specific application.

   b. The installation process should automatically create icons within the Wabi Application Manager. For more information on creating icons to launch Windows applications directly from your AIXwindows Desktop or Motif menus, see 7.4, “Wabi Desktop and Windows Integration” on page 59.

You can now run and use Wabi. You may wish at this time to set a system default font cache. For details on how to do this, see 7.1, “Fonts” on page 51.

3.2 Wabi Prerequisites

Before installing Wabi, first check that you have the prerequisite levels of AIX and AIXwindows installed on your systems.

3.2.1 AIX Level

You can check your AIX level by using the command:

```
# lslpp -L bos.obj
 Description                                          State     Fix Id
 ---------------------------------------------------  ------    ------------
 bos.obj 3.2.0.0
   3250 bos Maintenance Level                            C     U491123
   3250 AIX Maintenance Level                            A     U493250
     HFT Configuration Utilities                         C     U423532
     Kernel                                              C     U423980
```

```
     Device Drivers                                          C    U423981
```

Look at the number before the AIX Maintenance Level. If the number is 3250 you are on AIX level 3.2.5. This is the required level of AIX for running Wabi. The system shown above is running on level 3.2.5 with three additional updates beyond that level (for the HFT, Kernel and Device Drivers).

If the AIX Maintenance Level shows number 3240, you are using AIX Version 3.2.4. You must upgrade your system before installing or running Wabi.

If you get the following response:

```
# lslpp -L bos.obj
lslpp: illegal option -- L
Usage: lslpp {-A|-d|-f|-h|-i|-l|-p} [-B | -I] [-acJq] [-O{[r][s][u]}]
             [product ... | fix_id ... | all]
```

your system is running an AIX level below 3.2.4. You must upgrade your system before you can run Wabi.

3.2.2 AIXwindows Level

You can check the level of AIXwindows that is on your system with the following command: (Note the capital X)

```
# lslpp -L X11rte.obj
 Description                                          State     Fix Id
 ---------------------------------------------------  ------    ------------
 X11rte.obj 1.2.3.0
   3250 X11rte X11-R5 Maintenance Level                   A    U491119
     AIXwindows Run Time Environment                      A    U409194
     AIXwindows Run Time Environment                      A    U411705
```

Wabi 1.1 is supported by IBM on AIXwindows Environment/6000 1.2.5. This corresponds with release 5 of the X Window System (X11R5). In the listing above, the level of AIXwindows appears as 1.2.3. Version 1.2.5 is a set of fixes on top of version 1.2.3. If the system has the 3250 X11rte X11-R5 Maintenance Level applied as shown above, then the system is at the correct level.

3.2.3 AIXwindows Fixes for GT Graphics Adapters

If you are using a system with a GT3, GT3i, GT4e, GT4, GT4x, GT4i or GT4xi graphics adapter, you should ensure that you have applied the Program Temporary Fix (PTF) U424296. To determine if you have this PTF installed, execute the command:

```
# lslpp -hB U424296
```

If the required fix is installed, the output will appear similar to:

```
  Name
  --------------------
    Fix Id  Release         Status     Action     Date       Time     User Name
    ------- --------------- ---------- ---------- ---------- -------- ----------
  bth: /usr/lib/objrepos
  X11rte.obj
    U424296 01.02.0003.0000 COMPLETE   APPLY      02/15/94   11:59:54 root
```

```
Path: /etc/objrepos
  X11rte.obj
    U424296 01.02.0003.0000 COMPLETE   APPLY      02/15/94   12:00:23 root
```

If the fix is not installed, output will be similar to:

```
lslpp:  There is no fix id in /usr/lib/objrepos, /etc/objrepos,
        or /usr/share/lib/objrepos that matches "U424296".
```

If required, contact your local AIX Software Support Center to obtain the PTF.

3.3 Installing Wabi

Wabi is installed in the same manner as any other IBM AIX/6000 software product. The steps are as follows:

1. Insert the installation media in the drive.
2. As the root user, start the System Management Interface Tool (SMIT) using the command:

   ```
   # smit install_latest
   ```

   Press the **F4** key and select your installation media. You will then see the screen shown in Figure 5:

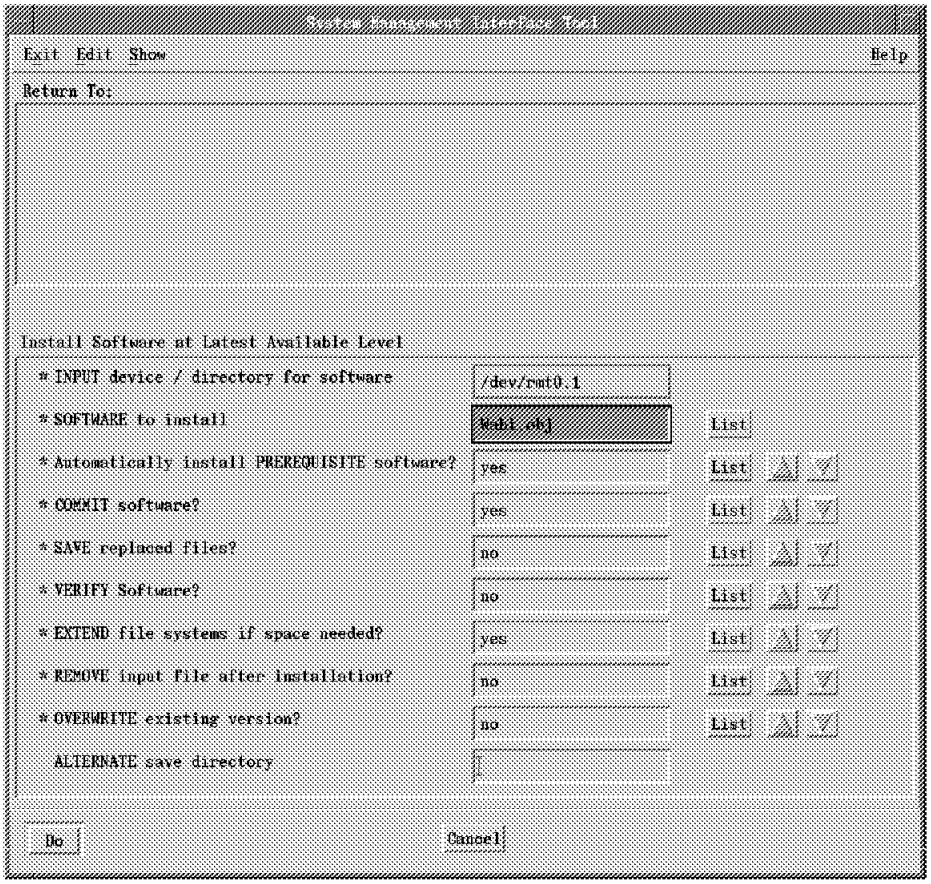

Figure 5. SMIT Installation Panel

3. The only field to change on this screen is the SOFTWARE to install. You can either type **Wabi.obj** or press the **List** button and select **Wabi.obj** from the Table of Contents.

   Click on the **Do** button to install Wabi.

   For more information on using SMIT to install software, see the AIX Version 3.2 for RISC System/6000 Installation Guide.

4. The directory /usr/lpp/Wabi/bin should be added to the search path of all Wabi users. This can be done by editing the user's .profile file using your favorite text editor. A standard, unmodified PATH variable for an AIX user is:

   ```
   PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:$HOME/bin:/usr/bin/X11:/sbin:.
   ```

   To add Wabi to the search path, modify this to be:

   ```
   PATH=/usr/bin:/etc:/usr/sbin:/usr/ucb:
   $HOME/bin:/usr/bin/X11:/sbin:/usr/lpp/Wabi/bin:.
   ```

   The user will need to log off and log back on to get the new search path.

5. If you use the AIXwindows Desktop, run the following command after after starting up the desktop:

   ```
   $ /usr/lpp/Wabi/bin/xdt3wabi
   ```

   This will create a Wabi icon on the desktop for automatically launching Wabi. For more information on integration of Wabi into the AIXwindows Desktop, see 7.4.1, “AIX Desktop” on page 60.

6. You can now start Wabi by typing the command:

   ```
   $ wabi
   ```

   Note: You must already be running AIXwindows to be able to run Wabi.

7. You may also now wish to set a system default font cache. For details on setting the default font cache, see 7.1, “Fonts” on page 51.

8. You should ensure that you obtain the most recent information on Wabi 1.1 by reading the release notes that accompany the program. You can read these notes by double-clicking on the **Wabi 1.1 Release Notes** in the Tools group.

3.4 Installing Microsoft Windows 3.1

It is not necessary to install Microsoft Windows 3.1 to run Windows applications under Wabi. There are three main reasons for installing Microsoft Windows 3.1:

- Wabi 1.1 contains native implementations of several Windows Dynamically Linked Libraries (DLLs). Other DLLs are not implemented natively and will run under the Wabi 386 instruction emulator. These non-native DLLs are not supplied with Wabi. Some of these DLLs are supplied with Windows applications. Otherwise, the DLLs must be obtained by installing Microsoft Windows 3.1. Thus, if your application requires one of these non-native DLLs, and does not include the DLL, you must install Microsoft Windows 3.1 or another application that includes the DLL. Of the thirteen applications that are qualified to run under Wabi 1.1, only two require that Microsoft Windows 3.1 be installed:
  - Microsoft Powerpoint
  - Paradox

- Part of the appeal of Microsoft Windows 3.1 is the range of small applications, or applets, that it includes. These applets include Write, Paintbrush, Calculator, Terminal, Notepad and File Manager. Wabi 1.1 does not include these applets. If you require these applets, you should also install Microsoft Windows 3.1.
- Microsoft Windows 3.1 includes standard help functions that are used by both Windows itself and by many Windows applications. These help functions will not operate without installing Windows since the program WINHELP.EXE is not supplied as part of Wabi.

For experienced Windows users, it is important to note that you **do not install Microsoft Windows 3.1 under Wabi by running setup from the A: drive.** Because Wabi uses some of the original Windows DLLs and in other cases natively implements the DLLs as native RISC System/6000 executable code, the installation process must be careful to selectively extract files from the Microsoft Windows 3.1 distribution media, while inserting the appropriate files from the Wabi distribution.

To install Microsoft Windows 3.1 under Wabi, perform the following steps:

1. Log in as your Wabi user and start Wabi by typing the command:

   ```
   $ wabi
   ```

2. Double-click on the **Windows Install** icon. You will now see the following dialog box.

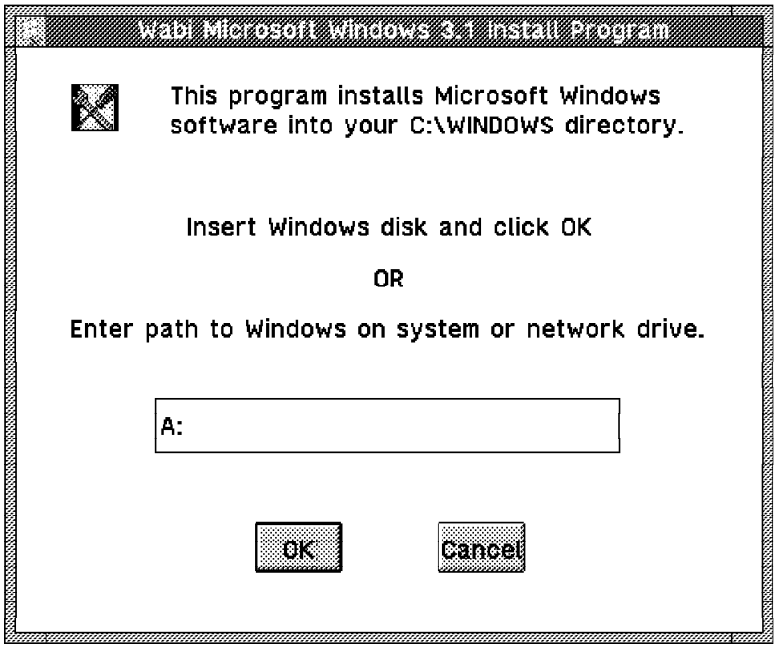

Figure 6. Install Windows Panel

3. Insert the first Microsoft Windows 3.1 diskette into the diskette drive and click on **OK**. For information on installing Windows from an installation image on hard disk, see 3.6, “Copying Installation Files to Disk” on page 24.
4. Follow the instructions and insert the remaining Microsoft Windows 3.1 diskettes when requested.
5. You have now installed Microsoft Windows 3.1 and can rearrange the application groups as required.

3.5 Installing Applications

You have now reached the stage to install your Windows applications to run under Wabi. In general, this is performed in exactly the same way as installing the application under native Microsoft Windows 3.1

If you are installing the same applications for several users, there are much quicker and simpler ways to do it than changing diskettes each time. See 3.6, “Copying Installation Files to Disk” on page 24 for details.

The steps for installing a Windows application are as follows:

1. Click the **Run** option from the File pull-down menu and enter the applications install command. Figure 7 is an example of a completed run dialog box.

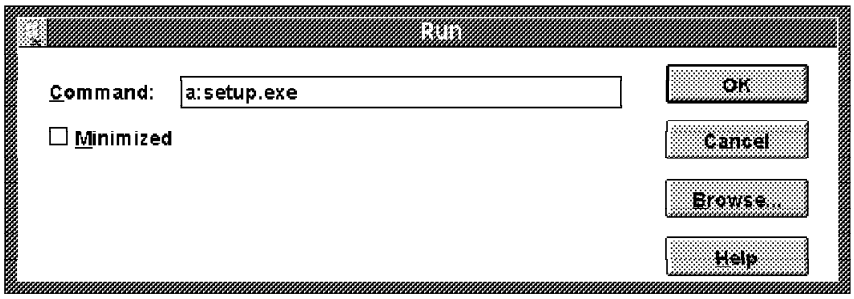

Figure 7. Run Dialog Box

2. In most cases, the application's installation command will be called SETUP.EXE or INSTALL.EXE but this may not be the case. You can look at the contents of the diskette by clicking on the **Browse** button. This will give you the dialog box shown in Figure 8.

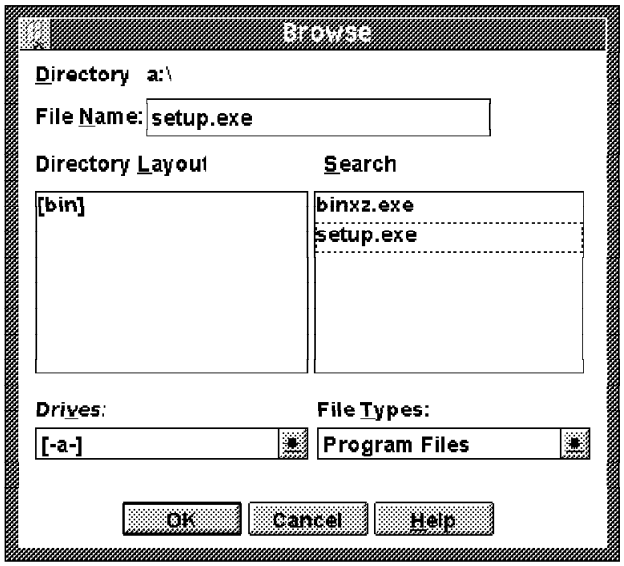

Figure 8. Browse Dialog Box

From this dialog box, you can select the required installation program from the diskette. Usually, the name will be obvious. After typing or selecting the installation program, click on **OK** in the Run dialog box.

3. For details on the installation options for the application you are installing, see the application's installation guide.

3.6 Copying Installation Files to Disk

If you have to install Microsoft Windows 3.1 for many users, or have to install certain Windows applications multiple times, it is much easier to copy the files from the installation diskettes to a single installation directory.

There are two ways to do this step:

- Create a directory under Windows and use the Windows File Manager to copy the files from the diskette to the hard disk.
- Use dosread to copy the files under AIX.

  We wrote a shell script called fd2hd that will ask for the name of a directory to create and the number of installation diskettes to copy, then will prompt you to insert each diskette in turn and copy the files to disk. The shell script is included in Appendix C, “Redbook Sample Tools and Utility Programs” on page 111.

You will find that the files copy more quickly using the first method, but it is easier to use the second method as you don't need to select all the files on each diskette and drag them onto the disk - you just change diskettes when instructed.

This procedure has not been tested with all of the qualified applications but has worked for every case we have tried. This procedure in no way effects the licensing of your Windows applications. You will still require an individual or site license for each system where the application is installed.

Note: This script only copies the product's installation images to the hard disk. You will still need to run the product's install program to install the product on the system

Chapter 4. Application Support

In this chapter, we describe:

- Which Windows applications have been qualified
- Which Windows applications are not supported
- Our experiences using some unqualified applications
- The steps required to move applications to a different hard drive
- How to use the Windows 3.1 Program Manager instead of the Wabi Application Manager
- How to transfer text files between DOS and UNIX

4.1 Qualified Applications

The following applications are qualified by IBM as being compatible with Wabi 1.1. This means they have been tested and all functions operate in the same way as when running under native Windows.

- Word Processors
  - Microsoft Word for Windows 2.0
  - WordPerfect** for Windows 5.2
  - Lotus AmiPro** 3.0
- Spreadsheets
  - Microsoft Excel** 4.0
  - Lotus 1-2-3** for Windows 1.1
  - Borland Quattro Pro** for Windows
- Databases
  - Borland Paradox for Windows (requires Microsoft Windows 3.1 to be installed)
- Presentation Graphics
  - Microsoft PowerPoint 3.0 (requires Microsoft Windows 3.1 to be installed)
  - Harvard Graphics** for Windows
- Graphics Tools
  - CorelDRAW** 3.0
- Project Management
  - Microsoft Project** 3.0
- Desktop Publishing
  - Aldus PageMaker** 4.0
- Communications
  - PROCOMM PLUS** for Windows 1.0

Other applications that conform to Microsoft Windows 3.1 application program interface (Win16 API) conventions may operate correctly under Wabi 1.1. Lack of inclusion in this list does not mean that the application will not work - only that it has not been fully tested.

4.2 Non-Supported Applications

The following applications are *not* supported under Wabi 1.1

- Adobe Type Manager** or applications that use it.
- Network applications or a Windows application that uses network facilities such as:
  - Netware** requests
  - WinSock requests
  - NetBios requests
- Applications that use the Multi-Media Extentions

4.3 Non-Qualified Applications

There is another class of applications that are not listed as qualified applications and are also not listed as not supported. These applications are called non-qualified. In other words they may or may not work depending on how they interact with the Windows and DOS facilities. We tried out a few of these and here is what we found.

Note: These have been installed under a beta version of Wabi, Windows version 3.1 and tested informally by IBM staff or customers. Some notes on the levels of success are included where available. Please note that this information has not been subjected to any formal review process and is presented as a guideline only. You should test any applications personally before deciding whether or not Wabi is suitable for running a particular application.

4.3.1.1 AfterDark 2.0

The basic AfterDark** display functions appear to operate correctly, however we were not able to set a password as we could not type in the password panel. The Flying Toasters option failed to display due to lack of memory however there may be configuration options to get around this. Most importantly, the Aquatic Realm appears to display correctly. AfterDark gives an error on starting that it cannot access a sound device. AfterDark also causes an unrecoverable error in the application when you exit, caused by an Exception D.

AfterDark will cover your entire screen - both Windows and AIX applications. Be aware that if you are working with AIX applications, AfterDark will detect that Windows or Wabi is idle and will still lock up the entire screen.

4.3.1.2 FrameMaker 3.0

FrameMaker** 3.0 does not currently run under Wabi. It fails when starting with an Unrecoverable Error in Application caused by an Exception D. You will then have to restart your Wabi session before you can continue operation.

4.3.1.3 Windows Entertainment Pack

All programs appear to work correctly except for Pegged (PEGGED.EXE). Pegged works to some degree, but has a problem where a peg will disappear as another peg is moved towards it. The peg can be seen by refreshing the display.

4.3.1.4 Windows Entertainment Pack Two

All programs appear to operate correctly except for Rodent's Revenge (RODENT.EXE) and RattlerRace (RATTLER.EXE). Both of these fail with the error "File Not Found".

4.3.1.5 Windows Entertainment Pack Three

All programs appear to operate correctly except for TetraVex (TETRAVEX.EXE), which fails with an unrecoverable error in the application caused by an internal error: Native Error 11, running TETRAVEX.

4.3.1.6 WordPerfect for Windows 6.0

This will not install under Wabi 1.1.

4.4 Moving Applications

We do not recommend moving applications from one hard drive letter to another (for example, from the C: drive to the D: drive). A safer procedure is to re-install the application from the installation media or from a hard disk copy of the installation media. There are three reasons for this:

- When an application is installed, the installation process can place files in several different areas. It can be difficult to ensure that you have found all the files that need to be moved.
- The system and application initialization files (with a .ini extension) contain information that is hard-coded to the location of the program.
- The actions to be taken when clicking on the application icons that appear under the shell program (Application Manager or Program Manager) are also hard coded to the location of the program. The actions to be taken for these icons are contained in group files (with a .GRP extension) and the groups are also listed in the progman.ini file.

If you do wish to try to move an application, the .ini files are plain text and can be edited using a Windows text editor. However, the .GRP files are stored as binary data and cannot be edited by hand.

When you move the application files, the icons for that application will change to the generic Wabi icon and will no longer operate. You will have to recreate these icons by hand.

To re-create an icon:

1. Select the application item whose parameters you want to change.
2. Choose **Parameters** from the **File** menu.
3. In the Application Parameters dialog (see Figure 9 on page 28) , enter the new paths for the **Command** and **Working Directory**. Alternatively, you you can use the PATH variable specified in your AUTOEXEC.BAT file to locate the executables.

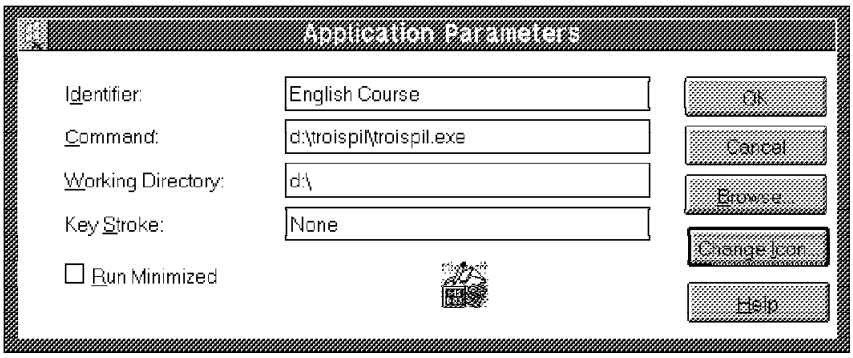

Figure 9. Application Parameters Dialog Box

4. Click on the **Change Icon** button and change the path in the **File** field.
5. Click **OK** in the Application Parameters dialog to save your changes.

4.5 Changing the Windows Shell

One application of particular importance to Windows is the shell program. The main function of the shell program is to provide an interface from which you can launch other applications. The shell program is the initial interface that Windows supplies to the user and is used to launch other applications. When you exit the shell, Windows assumes that you are finished and closes down Windows.

The following are examples of shell programs:

- PROGMAN.EXE
- APPMAN.EXE

The shell program is specified to Windows in an entry near the the top of the system.ini file. The entry will look like this:

```
shell=appman.exe
```

You can change this line to specify any Windows program. Remember, since the shell is the interface that is intended to launch other applications, specifying a shell without launch capabilities could produce undesirable results. Regardless of which program you specify it must reside on your C: drive or be linked to your C: drive.

The default shell under Wabi is the Application Manager, APPMAN.EXE. During Wabi installation, two links are created in your $HOME/wabi/windows directory These links are:

- The APPMAN.EXE shell which is a link to the executable program APPMAN.EXE located in /usr/lpp/Wabi/wbin directory.
- The PROGMAN.EXE shell which is also a link to the executable program APPMAN.EXE located in /usr/lpp/Wabi/wbin directory.

Since these names link to the same program, changing the shell specification in the system.ini file to use PROGMAN.EXE in place of APPMAN.EXE will have no effect. However, if you install Microsoft Windows 3.1 the PROGMAN.EXE link will be replaced by a separate program, giving you a second shell program to choose from.

There are several reasons why you may wish to specify the PROGMAN.EXE shell instead of APPMAN.EXE, including:

- You want to use the restrictions options of the progman.ini file to modify some of the settings in the pull-down menus of the Program Manager (see section 8.2.4.1, “Controlling the Program Manager Pull-Down Menus” on page 75 for more information).
- If you have installed a version of Microsoft Windows 3.1 in a language other than English, the pull-down menus under the Application Manager will be incorrectly shown in English. The Program Manager displays these menus in the correct language. Note that in either case, the window pull-down menu shown when you click on the window's close button will still appear in English under Wabi 1.1.
- Your Windows application may require the PROGMAN.EXE shell.

4.6 Moving Files between Wabi and AIX

AIX and DOS (or Windows) use different characters for marking the end of each line in ASCII text files. AIX uses the Line Feed character (ASCII value 10) to mark the end of a line. DOS uses a Carriage Return character (ASCII 13) followed by a New Line character. Thus when you view a DOS text file such as the AUTOEXEC.BAT file under AIX, the Carriage Return character is not used to mark the end of line, and thus is displayed by the editor as a Control M character, represented as ^M.

If you display the file using the AIX `cat` or `pg` commands, you will not see these characters. Also, DOS uses a Control Z character (ASCII 26) to terminate a text file, while AIX uses a null character (ASCII 0).

You can convert a file between DOS (Windows) and AIX formats by using the AIX commands `unix2dos` and `dos2unix` that are supplied with Wabi in the directory /usr/lpp/Wabi/bin.

The format for these commands are as follows:

```
unix2dos unixfilename [ dosfilename ]

dos2unix dosfilename [ unixfilename ]
```

Where `unixfilename` is the name of the AIX format file and `dosfilename` is the name of the DOS format file. If the second filename is not specified, the original file will be replaced by the translated file.

Note: These programs should only be used to convert text files. They will not convert DOS programs to AIX programs. Binary data files that are not stored as text should not require any conversion to be transferred between DOS and AIX systems.

Chapter 5. DOS Emulation

Wabi 1.1 doesn't include a DOS emulator, however it can bring up a DOS session or start DOS applications after you have installed a separate DOS emulator program such as IBM's AIX Personal Computer Simulator/6000 (PCSIM).

In this chapter, we will describe:

- How to access a DOS session or DOS application from inside Wabi
- How to install and connect a DOS emulator to Wabi
- Using AIX Personal Computer Simulator/6000 as the DOS emulator
  - How to setup and use the AIX Personal Computer Simulator/6000
  - Some of the problems that can occur when using AIX Personal Computer Simulator/6000
  - How to avoid these problems while installing AIX Personal Computer Simulator/6000

5.1 How to Access a DOS Session or DOS Application from Inside Wabi

A DOS emulator must be installed and connected to Wabi before you can bring up a DOS session or a DOS application under Wabi. See 5.2, “How to Install and Connect a DOS Emulator to Wabi” on page 32 for information on how to install and connect an emulator to Wabi.

Once the emulator has been installed and connected to Wabi, a DOS session can be started from within Wabi by either of these two methods:

- Clicking on the **MS-DOS Prompt** icon of Windows Main panel
- Clicking on the **DOS Session** icon on Wabi Tools panel

For DOS applications, Wabi detects that the executable is a DOS application and not a Windows application so it starts a DOS emulator to execute the application. The name of the executable and any parameters that you specify can be passed as command line arguments to the emulator at start up. These arguments are typically used to have the emulator immediately start the desired application after the emulator initializes. See 5.2, “How to Install and Connect a DOS Emulator to Wabi” on page 32 for more information on specifying arguments on the emulator start up command.

Note: Not all emulators support command line arguments that specify an executable to bring up after initialization. The AIX Personal Computer Simulator/6000 is an example of an emulator that does *not* support this method for starting applications in the emulator. See 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36 for information on how to perform this function with the AIX Personal Computer Simulator/6000 product.

DOS applications can be started from within Wabi by one of these three methods:

- Double clicking on the application's icon in the Windows File Manager.

- Entering the name of a DOS application and optional parameters into the command field of the Run dialog box. This dialog can be accessed from either the Wabi Application Manager pull-down menu or Windows File Manager pull-down menu.
- Creating your own Application item and icon that specifies your DOS application and optional parameters as the command to start up. Once created, the Application can be started by clicking on the icon that you created for the application.

5.2 How to Install and Connect a DOS Emulator to Wabi

See the installation instructions of your particular emulator for information on how to install the emulator. If you have chosen the AIX Personal Computer Simulator/6000 product as your emulator, see 5.3, “Using AIX Personal Computer Simulator/6000 as the DOS Emulator” on page 33 for information that supplements the installation instructions for the AIX Personal Computer Simulator/6000 product.

To connect the installed DOS emulator to Wabi:

1. Click on the **DOS Session** icon on the Wabi Configuration Manager panel to bring up the DOS Emulator Connection dialog box
2. Enter into the DOS Emulator Command field, the command string for starting the DOS emulator. See Figure 10 for an example of this completed dialog box.

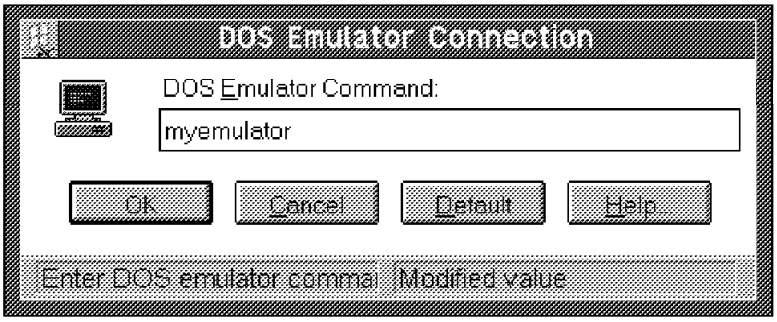

Figure 10. DOS Emulator Connection Dialog Box

If you wish to start up a DOS application using any of the methods listed in 5.1, “How to Access a DOS Session or DOS Application from Inside Wabi” on page 31, you may need to add the following tokens to modify the command that Wabi uses when starting the DOS emulator. See 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36 for information on how to do this is you are using the AIX Personal Computer Simulator/6000.

%d will be replaced by the remote display name that is currently being used to display Wabi. This token should only be used if you desire to display the DOS emulator on the same remote display as Wabi. Local displays can omit this token

%f will be replaced by the name of DOS executable (.EXE file) that is attached to the icon that was doubled clicked or specified using the RUN or Application Item facilities mentioned above

%c will be replaced by any specified parameters to the DOS executable. For example, if you start up a DOS application using the Application Manager RUN facility, and specify parameters on the command line, the %c will be replaced by those parameters in the DOS emulator command

A DOS emulator start up command incorporating these tokens could look like:

```
myemulator -display %d -c %f %c
```

In the above example, myemulator is started on the same remote display as Wabi is currently using. The specified DOS application will be immediately started with any passed command line arguments.

If you were then to create a Windows Application Item as shown in Figure 11:

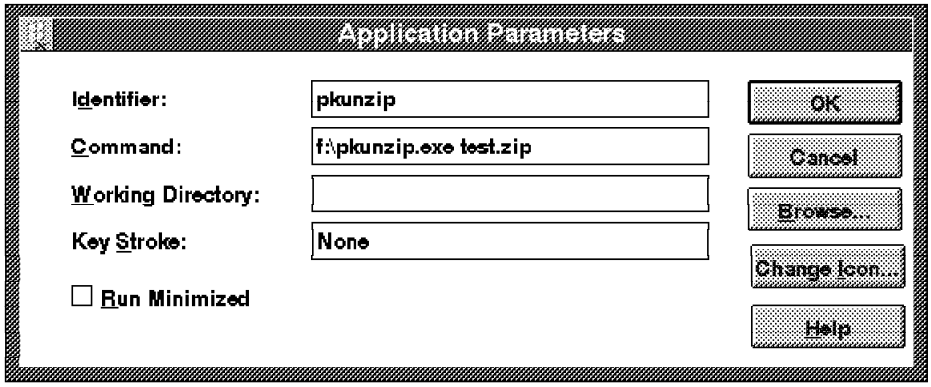

Figure 11. Sample New Application Item

Your DOS application would then be invoked as:

```
myemulator -display <Display Name> -c f:\pkunzip.exe test.zip
```

5.3 Using AIX Personal Computer Simulator/6000 as the DOS Emulator

AIX Personal Computer Simulator/6000 is designed so users can run IBM Disk Operating System (DOS) application programs on an IBM RISC System/6000 in an AIX environment. Details on the basic installation and setup of the PC Simulator are given in the PC Simulator/6000 Guide and Reference. There are four additional points that you must be aware of when using the PC Simulator under Wabi:

- Setting up your C: drive
- Drive/Path mappings
- DOS filenames
- Running DOS applications from a Wabi icon

5.3.1 Setting Up the C: Drive

The PC Simulator can use either an AIX data file or an AIX directory as an emulated DOS disk drive. If an AIX directory is used to emulate a DOS disk drive, the DOS files that you save on the DOS disk drive are actually stored as separate files in the AIX directory. If an AIX file is used to emulate a DOS disk drive, the stored DOS files are actually encoded into that single AIX file.

The recommended (by the PC Simulator documentation) normal configuration for the PC Simulator is to use a single file for the C: drive and directories for other drives. Unusual side effects can result if you use a directory as a C: drive. For instance, the DOS `append` command is not currently supported when a directory is used as the C: drive. See Using DOS on PC Simulator in the IBM AIX Version 3.2 for RISC System/6000 General Concepts and Procedures for more information.

If you elect to use the normal configuration for disk drive setup under the PC Simulator, you will not be able to use Wabi or AIX to see, access or modify any of the files that you store on to your PC Simulator C: drive. This is because all of the individual DOS files on the drive are encoded into a single AIX file. You will also not be able to use the workaround described in 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36 to have your DOS application automatically started in the PC Simulator when you click on the associated Wabi icon. The workaround, as you will see, requires that you modify the AUTOEXEC.BAT file that the PC Simulator uses at start up. The AUTOEXEC.BAT is stored on the C: drive which is encoded into a single AIX file and not editable from AIX. If you use a directory for the C: drive as the workaround suggests, you can automatically execute your application as desired, but keep in mind that some functions of the PC Simulator are not supported in this alternate C: drive configuration.

5.3.2 Drive/Path Mappings

When you execute a DOS application under Wabi, Wabi looks at the executable file and determines that it is a DOS executable. Wabi then starts your DOS emulator using the command specified in the DOS Emulator Connection panel, and passes as one of the arguments to the DOS emulator, the path to the desired executable (see 5.2, “How to Install and Connect a DOS Emulator to Wabi” on page 32). The disk drive letter and path used to reference the executable is referred to as a drive/path mapping. The drive/path mapping must be the same between Wabi and the AIX Personal Computer Simulator/6000. If the drive/path mappings are different, the DOS emulator will not be able to access the executable as specified by the Wabi drive/path. To eliminate this type of mismatch, set up your PC Simulator as follows:

- Set up the C: drive as either a single file or a directory. See 5.3.1, “Setting Up the C: Drive” for more information on this choice.
- Install only DOS on the C: drive.
- Set up additional disk drives (D:, E:, and others) as directories in the AIX filesystem. It is sometimes helpful to name the AIX directories something like dosdiskd or dosdiske for the D: and E: drives respectively. This naming convention will help you remember what these directories are for and more importantly which directory is associated to which drive.
- Install your DOS applications only on these additional disk drives.
- Connect these AIX directories (DOS disk drives) to Wabi disk drives. Remember to be sure and use the same drive letter in Wabi as you used in the

PC Simulator. Figure 12 on page 35 shows how a file in AIX can be accessed in both the AIX Personal Computer Simulator/6000 and Wabi using the same drive/path mapping

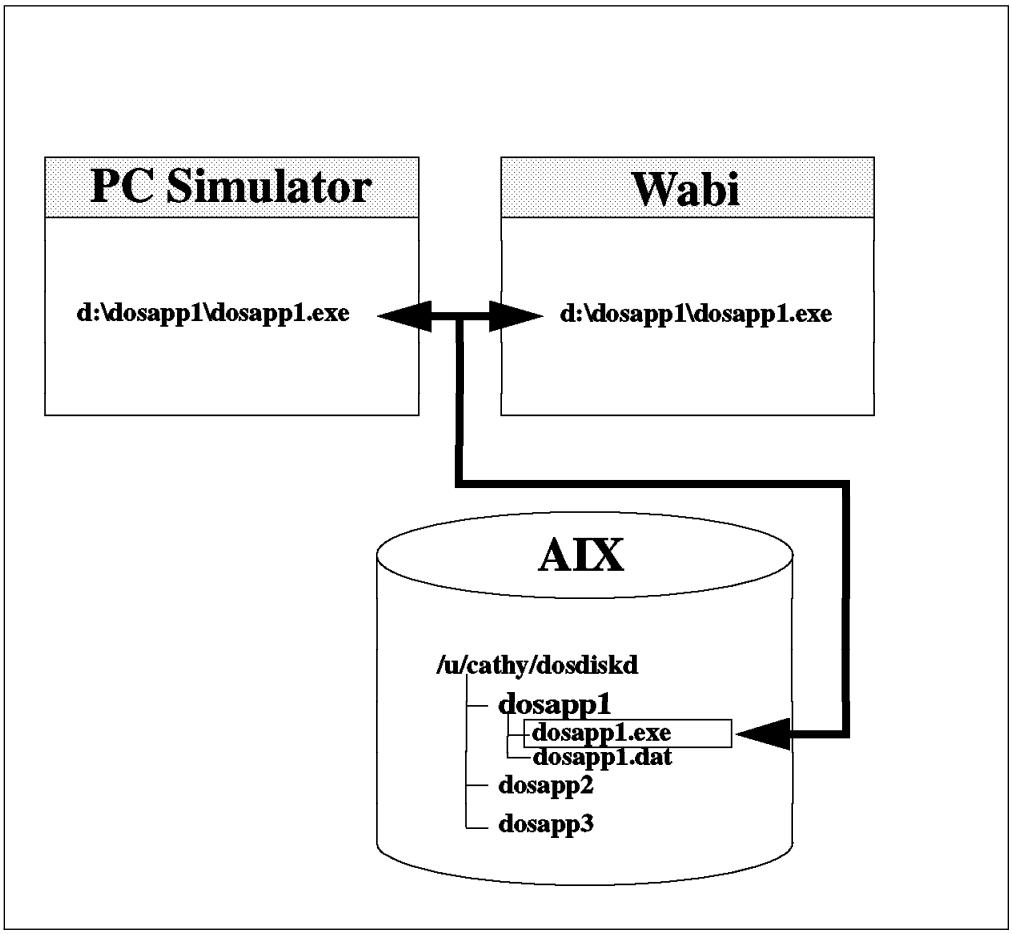

Figure 12. Accessing the Same File from the PC Simulator and Wabi

See 6.2, “Disk Drives” on page 43 for more information about connecting directories to Wabi disk drives. Some drives letters in Wabi are reserved and can't be changed to attach to AIX directories. For example, the C: drive is connected to the $HOME/wabi directory, and can't be used for anything else. Other reserved drive letters are R: and W:.

5.3.3 DOS Filenames

When creating or copying files from diskette using the PC Simulator with the default configuration, the filenames are written in uppercase letters under the AIX filesystem. Filenames with uppercase letters are not legal DOS filenames for Wabi.

Wabi uses the following conditions for a legal DOS filename:

- All lowercase letters
- Filename no longer than eight characters
- Extension no longer than three characters
- Any characters except the following ones . , +] [* ? : \ / ; = < >

When the filename isn't a legal DOS filename, Wabi translates the filename to a legal DOS filename. For instance, AUTOEXEC.BAT will be mapped in autoe~aa.

See 2.7, “Filename Translation” on page 12 for more information about file naming.

To avoid the translation when creating files with the PC Simulator, use the `case L` option of the `pcsim` command. This specifies that filenames created while using the PC Simulator are written in lowercase so that the files will be legal for both Wabi and the PC Simulator.

5.3.4 Running DOS Applications from a Wabi Icon

Under the PC Simulator command there is no way to specify on the pcsim command line the name of a DOS program to run after initialization. Thus, there is no way to use the Wabi provided tokens (`%f %c`) on the simulator's start up command. One way to work around this problem is to use our shell script and C program that we wrote to update the AUTOEXEC.BAT that the PC Simulator will use at start up. The PC Simulator C: drive must be defined as a directory in order for this workaround to function. See 5.3.1, “Setting Up the C: Drive” on page 34 for more information.

The shell script we created is called dosexe and takes in two parameters:

- The full path name of your DOS executable file
- Any optional parameters you which to supply to the DOS executable

The shell script along with the AIX executable, copies the original AUTOEXEC.BAT file to an AUTOEXEC.SAV file, then it modifies the AUTOEXEC.BAT by adding the following three lines:

```
<drive>:
cd <full path of your DOS application>
<your DOS executable file> <parameters>
```

Finally, it runs the `pcsim` command.

You can specify this shell script as the DOS emulator in the DOS Emulator Connection dialog box (see section 5.2, “How to Install and Connect a DOS Emulator to Wabi” on page 32). An example of an emulator start up command using our workaround would be:

```
dosexe '%f' '%c'
```

For the contents of the shell script and C source code for the executable, see C.5, “Sample Tool/Utility: dosexe” on page 115.

5.3.5 AIX Personal Computer Simulator/6000 Installation and Setup Example

Here is an example of the installation and setup for the PC Simulator when using it with Wabi and the workaround mentioned in 5.3.4, “Running DOS Applications from a Wabi Icon.”

1. Install the Licensed Program Product (LPP) AIX Personal Computer Simulator/6000 as directed by the product installation instructions.
2. Create an AIX directory that will be used as the DOS C: drive.

   ```
   mkdir /u/cathy/dosdiskc
   ```

3. Create an AIX directory that will be used as the DOS D: drive.

   ```
   mkdir /u/cathy/dosdiskd
   ```

4. Insert the DOS diskette into the diskette drive on the machine.
5. Start up the PC Simulator from your home directory in AIX with the following command.

   ```
   pcsim -A 3 -C /u/cathy/dosdiskc -D /u/cathy/dosdiskd -dmode V \
   -case L -lpt1 asc -refresh 50 -xmemory 8192 -mouse com1 -save
   ```

   This command will start up the PC Simulator and save the configuration into a file called simprof. The contents of this file can be found in C.4, “Sample Tool/Utility: simprof” on page 114. C: drive (`setup` command or `fdisk` command).
6. Install DOS on the C: drive using the installation instructions that came with your version of DOS.
7. Stop PC Simulator by pressing the **Esc** key and typing **pcsim**.
8. Remove the diskette from the diskette drive
9. Restart the PC Simulator with the following command:

   ```
   pcsim
   ```

10. Install any desired DOS applications on the D: drive.
11. Stop the PC Simulator by pressing the **Esc** key and typing **pcsim**.
12. Start up Wabi.
13. Set up the DOS emulator in the DOS Emulator Connection dialog box to call the dosexe shell script (Figure 13) using the full path name to our shell script. See 5.2, “How to Install and Connect a DOS Emulator to Wabi” on page 32 for more information.

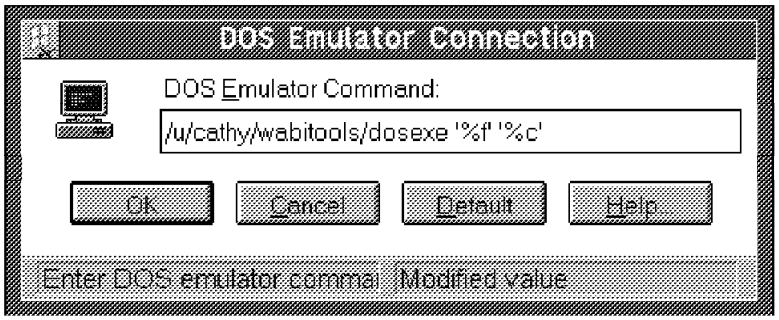

Figure 13. DOS Emulator Connection Dialog Box

   See 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36 for more information on this shell script.
14. Create icons in any Windows application group for your DOS applications. See (Figure 11 on page 33) for an example.

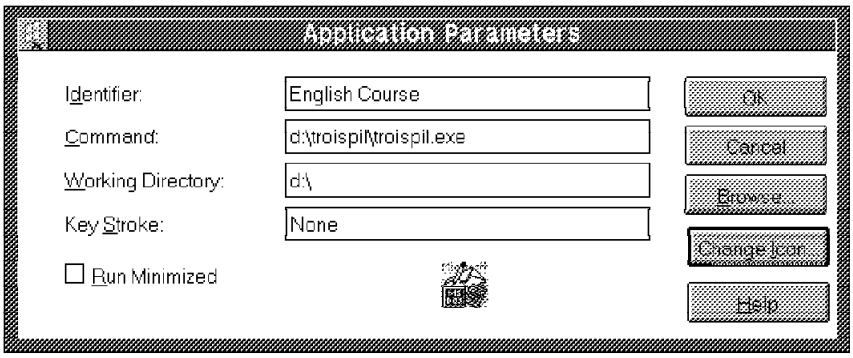

Figure 14. Application Parameters Dialog Box

15. Click on your new DOS application icons and verify that the DOS applications all execute as desired.

Chapter 6. Devices

Wabi provides access to AIX printer queues, local or remote files and directories, the CD-ROM drive and serial ports. In order to access and use any of these devices under Wabi, they must first be defined and made available under the AIX operating system.

The configuration of these devices is performed using the icons in the Wabi Configuration Manager, shown in Figure 15.

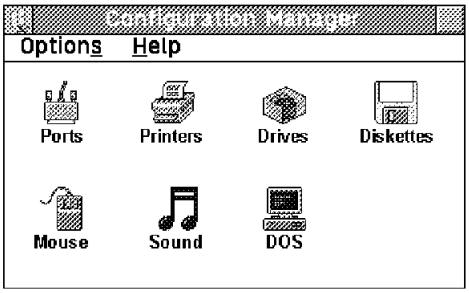

Figure 15. Wabi Configuration Manager Panel

The Configuration Manager provides functions similar to the Microsoft Windows 3.1 Control Panel. The Color option of the Windows Control Panel can still be used to set the Color Scheme for your Wabi installation, however the other functions of the Windows Control Panel will not operate or will cause errors. These functions should be performed through the Wabi Configuration Manager, or are not supported under Wabi at this time.

In this chapter, we describe:

- How to set up a printer in Wabi
- How to assign a drive letter to an AIX filesystem
- How to use the diskette drive
- How to use CD-ROM based applications
- How to set up serial devices

6.1 Printers

This section describes the four most important things to know about the Wabi printers:

- How to make a connection between a Wabi printer queue and an AIX printer queue
- Which printer drivers are supported
- The interaction between Wabi drivers and AIX drivers

6.1.1 Connection between AIX and Wabi

Wabi allows you to specify a connection between the various printer queues available under the AIX operating system and eight different Wabi printer ports. Serial connection information can also be specified for direct connections to serial connected printers. The eight Wabi printer ports are the following:

- Four serial printer ports (COM1 through COM4)
- Three parallel printer ports (LPT1,LPT2,LPT3)
- One file port for printing to a file

The connection between the printer and Wabi is established by either specifying an AIX printer queue and queuing command along with the desired Wabi printer port or the serial connection information for a direct connection to serial attached printers. For example, to use the asc printer queue defined under AIX as the LPT1 port under Wabi, specify to Wabi that LPT1 is associated with the asc queue and that print jobs can be sent to asc by issuing a `qprt` command.

This connection is specified to Wabi using one of three different ways.

Note: Option 1 may not be used to configure the print to file port.

1. Bring up the Wabi Configuration Manager and select the **Ports** option. The Port settings dialog box (Figure 16) is then displayed.

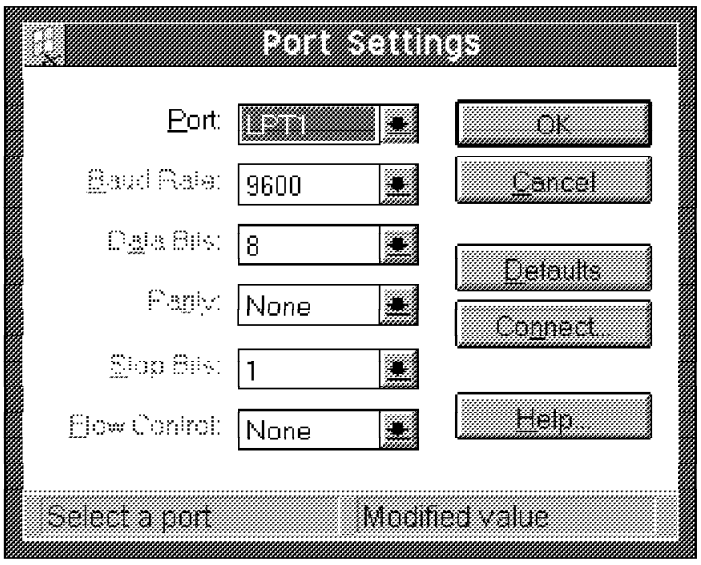

Figure 16. Port Settings Panel

In this dialog box select the desired Wabi printer port. If the port is a serial printer port, you can also specify the serial connection information in this dialog box. You can use the serial printer if you wish to bypass the AIX printer queues and send the print request directly to the serial attached device. Otherwise, you must specify an AIX printer queue to associate with this printer port. This can be done by pressing the **Connect** button. This action will bring up the Printer Output Connections dialog box (Figure 17 on page 41).

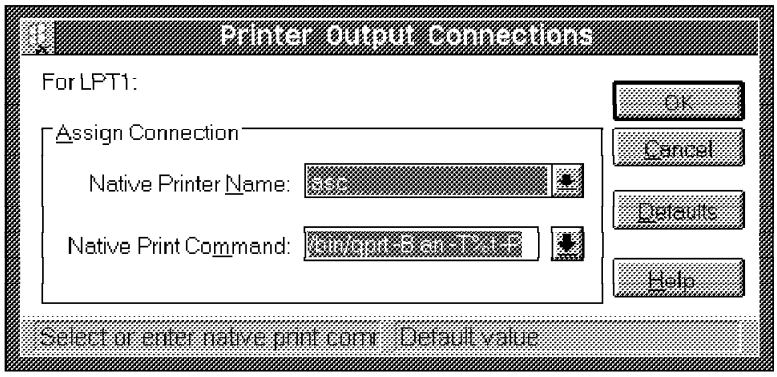

Figure 17. Printer Output Connections Dialog Box

This dialog box is used to specify which AIX printer queue (Native Printer Name) and which AIX queuing command (Native Print Command) Wabi should use when a print request is sent to the Wabi port specified in the Port settings dialog box. When finished supplying this information, press the **OK** buttons on the Printer Output Connections and Port Settings dialog boxes. The connection between the selected Wabi printer port and the AIX queue is now established and ready for use

2. Bring up the Wabi configuration Manager and select the **Printers** option. The Printer Settings dialog box is then displayed. From this dialog select the desired installed printer and press the **Port** button. This action will bring up the Printer Port Selection dialog box (Figure 18). From this dialog you can select the desired Wabi printer port and then press the **Connect** button to bring up and complete the Printer Output Connections dialog as above

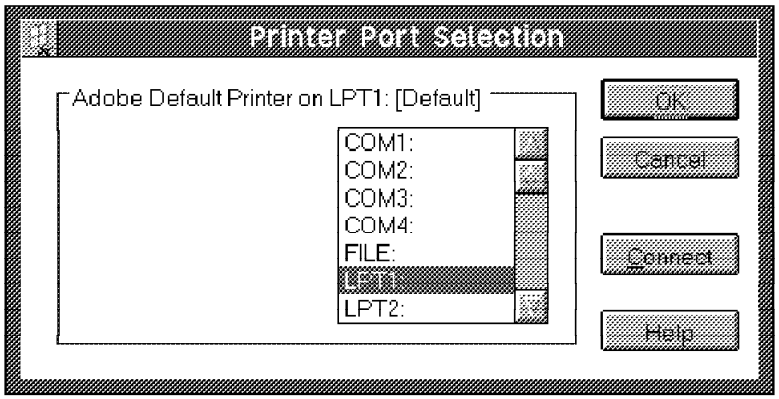

Figure 18. Printer Port Selection Panel

3. Use a text editor to modify the configuration file wabi.ini in your $HOME/wabi/windows directory. This method is not recommended due to the complexity of this configuration file, but is mentioned here for the following reason:

   If you have a lot of AIX queues defined on your system, you can have the system administrator define a default printer configuration for each user. Although, each individual user will still be able to change their own printer definitions, a default printer will have been preset for them.

Note: The wabi.ini file is a DOS file and *must only* be edited and modified with a DOS or a Windows editor such as the notepad applet. See Chapter 9, “wabi.ini File” on page 83 for more information.

Here is a sample of the printers settings in the [CommonSettings] section :

```
[CommonSettings]
Printers.command_lpt1=/bin/qprt -B an -T%t -P asc1
Printers.command_lpt2=/bin/qprt -B an -T%t -P ps
Printers.command_lpt3=/bin/qprt -B an -T%t -P asc2
```

- LPT1 is set to asc1 AIX queue
- LPT2 is set to ps AIX queue
- LPT3 is set to asc2 AIX queue

6.1.2 Printer Drivers

Wabi 1.1 supports all printers supported by the Adobe PostScript** printer driver. In addition, if you install the Microsoft Windows 3.1 software, you will also have access to a generic ASCII printer driver. This generic driver allows you to print text-only to a generic ASCII printer. No other printers are currently supported under Wabi 1.1.

You cannot install any printer drivers other than those listed in the Printer Settings dialog box list of Available Printer Drivers.

The Wabi 1.1 program does not support the downloading of fonts to a printer.

Driver software components of the Wabi product licensed by Adobe Systems Incorporated (Adobe) may only be used with printers containing PostScript software from Adobe. Adobe is a third party beneficiary to the license agreement which authorizes you to use the Wabi software with respect to the Driver Software components.

Printer drivers installed by Wabi are located in the /usr/lpp/Wabi/printers directory.

6.1.3 Interaction between Wabi Drivers and AIX Drivers

The default printer device driver in Wabi 1.1 is the Adobe PostScript printer driver. This driver will provide printer specific commands to do things like select paper source drawer, font handling, communications interfaces, and printer administration. Then, due to the connection between the Wabi port and the native queue, the PostScript file is sent to the AIX printer queue. Because the file sent by the Wabi device driver is a Postscript file, AIX print spooler passes the data stream to the printer with no change. The printer must be a PostScript printer able to interpret the PostScript instruction included in the file. If not, you will print the raw PostScript instructions included in the file.

When printing a non-PostScript file to a port linked to the Microsoft Windows 3.1 generic ASCII printer driver, the file is not modified by Wabi and is sent directly to the AIX ASCII printer queue connected to the Wabi printer port.

There is no interaction between the Windows application and the AIX printer driver. This means that the Windows application is not able to notify the user if the AIX queue is down, the printer needs paper or some other attention.

6.1.4 Wabi and the Microsoft Windows Print Manager

Wabi 1.1 does not support the use of the Microsoft Windows 3.1 Print Manager program.

Do not use Microsoft Windows Print Manager to install new printer drivers.

6.2 Disk Drives

Wabi allows you to access your AIX filesystems as drives represented by the letters C through Z. Any file accessible to the AIX filesystem can be accessed from a Wabi based application.

The files and directories you assign to drives need not reside on your local machine. If as you have adequate permissions, you can assign an NFS** filesystem or an AFS** filesystem to a Wabi drive.

To configure the disk drives, bring up the Wabi Configuration Manager and select the **Drives** icon. The Drive Connections dialog box is then displayed.

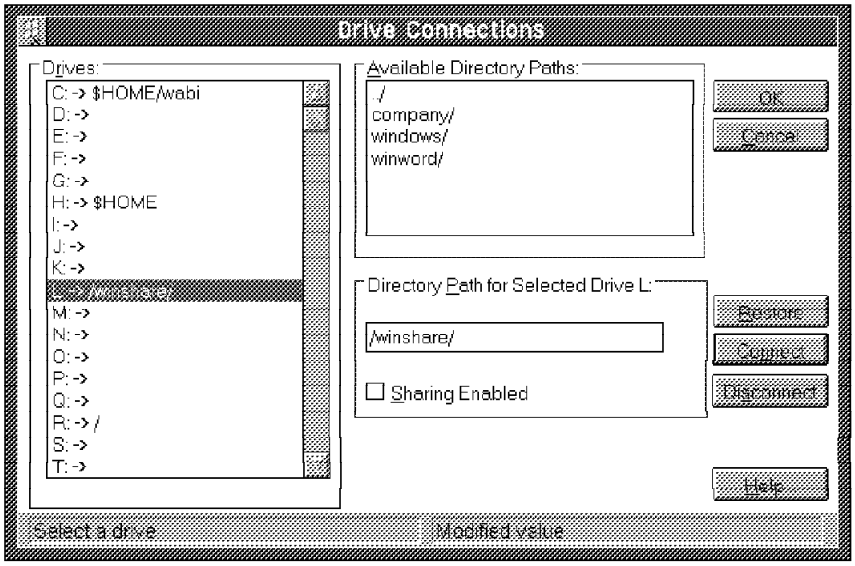

Figure 19. Drive Connections Dialog Box

The Drive Connections dialog box offers three panels:

- Drives panel lists the current drives assignments.

  Note: The C:, R: and W: drives are reserved at initialization. These reserved drives cannot be changed.

- Available Directory Paths panel displays the directories contained within the current directory. The highest level is the root (/) directory.
- Directory Path for Selected Drive panel indicates the path assignment of the selected drive.

The Drive Connections dialog box lets you do the following to the Wabi disk drives.

- Connect drives. The following procedure will assign an AIX directory path to a Wabi disk drive letter:

1. Select the desired drive letter to connect from the Drives panel.
2. Either select the desired AIX directory from the Available Directory Paths panel or enter the full path of the desired directory into the field in the Directory Path for Selected Drive panel.
3. Press the **Connect** button.

- Disconnect drives. To disconnect a non-reserved drive:
  1. Select the desired drive letter to disconnect from the Drives panel.
  2. Press the **Disconnect** button.
- Cancel current changes and start over by selecting the **Restore** button,
- Enable file sharing on the selected drive by selecting the **Sharing Enable** box in the Directory Path for the Selected Drive panel. See 8.3, “Sharing Data Files” on page 77 for more information on file sharing.

When making your drive connections, remember the following:

- Wabi has the following reserved drive assignments:
  - C: is connected to $HOME/wabi which points to the directory in which the Wabi program is installed. This directory can't be changed
  - R: is connected to / which points to the root directory. This directory serves as a gateway to network filesystems and can't be changed
  - W: is connected to $WABIHOME which points to a directory used to store Wabi files. This directory can't be changed
- Wabi has the following preset assignments:
  - E: is connected to $PWD which points to the current working directory when Wabi was started
  - H: is connected to $HOME which points to your home directory
- You must have adequate AIX permissions to access the files and directories you assign to the drives. If not, an error message indicating a permission problem will appear
- If you install a Windows application in a directory assigned to one drive letter, you will experience problems if you later re-assign the application directory to a different drive letter. For more information on this, see 4.4, “Moving Applications” on page 27.

6.3 Diskette Drives

This section shows you how to use a diskette drive under Wabi and the restriction on formatting diskettes.

6.3.1 Diskette Connections

Diskette drive A: is preset to AIX device /dev/fd0 file at initialization. If you want to use a second diskette drive, you must bring up the Wabi Configuration Manager and select the **Diskette** option. The Diskette Connections dialog box (Figure 20 on page 45) is then displayed and you can update the Diskette Drive B: field in the dialog box to indicate the AIX device that corresponds to the B: drive.

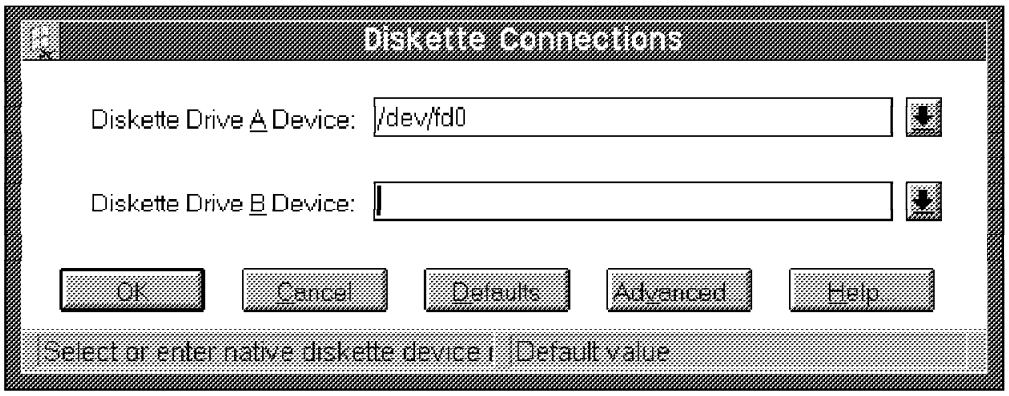

Figure 20. Diskette Connections Dialog Box

To aid you in your AIX device selection, this dialog box has a drop-down diskette drive list of the available diskette drive devices. This list is generated as a result of a device search using a set of searching parameters. In order to change the way Wabi searches for available diskette drives, you can modify these parameters in the Advanced Diskette Drive Options dialog box

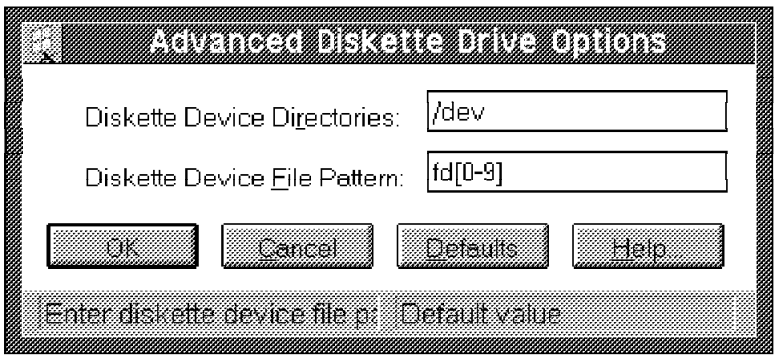

Figure 21. Advanced Diskette Drive Options Dialog Box

You can change both of these parameters in the Advanced Diskette Drive Options dialog box to alter the device search.

- The Diskette Device Directories parameter is a list of directories Wabi will search in to locate diskette drives. You can include more than one directory by using commas to separate directories names. Do not include spaces.
- The Diskette Device File Pattern is a regular expression which defines to Wabi a pattern for the filenames of the diskette device drivers. You can also include more than one pattern separated by commas without any intermediate spaces.

If you specify the Diskette Device File pattern with a * character, it will take a few minutes to bring up the Diskette Connections panel. Wabi is creating a list of all the files that exist under the /dev directory.

The default diskette devices and the searching parameters are stored in the wabi.ini file, [IBM-RS/6000] section.

```
[IBM-RS/6000]
Devices.disketteA=/dev/fd0
Devices.disketteB=
Search.FDDevicePathPattern=/dev
Search.FDDeviceFilePattern=fd[0-9]
```

See Chapter 9, “wabi.ini File” on page 83 for more information.

6.3.2 Formatting a Diskette

You can't format a DOS diskette in Wabi 1.1. You must use the AIX `dosformat` command or the DOS `format` command if you have installed a DOS emulator such as AIX Personal Computer Simulator/6000.

6.4 CD-ROM

Wabi lets you access CD-ROM devices that use the High Sierra filesystem (HSFS) format. Before assigning a drive to your CD-ROM, you must mount this device on your AIX system. Use `smit cdrom` command to configure the CD-ROM drive and then use the `smit fs` command to make a filesystem available for use at a specific location (the mount point). Then, bring up the Wabi Configuration Manager and select the drives option. Use the Drive Connections dialog box (Figure 19 on page 43) to assign a drive to this mount point.

See 6.2, “Disk Drives” on page 43 for more information about the Drive Connections dialog box.

Wabi 1.1 doesn't support multimedia extensions, so you can't use a CD-ROM drive to play music or video.

You may experience some problems if you use a CD-ROM based application that does not support using the application in a network environment. Some application vendors such as the Microsoft Corporation have incorporated a protection mechanism in their CD-ROM based application software that does not permit it to be run on a network drive. Although you may not be sharing the application as in a networked environment, because Wabi accesses your CD-ROM as network drive, it appears to the CD-ROM application that it is installed on a network and will not operate.

6.5 Serial Devices

Wabi allows you to connect to serial devices such as a serial printer or a modem.

See 6.1, “Printers” on page 39 for serial printer information.

Before you can use these devices on a COM port, you must:

1. Configure the settings on your hardware device (a modem for example)
2. Configure your serial devices in AIX (such as a tty)
3. Configure the COM port in the Wabi program

Be careful when configuring these parameters because the settings you choose must match between the device, AIX and Wabi. For example, if you set your modem to 9600 baud, the AIX tty that the modem is attached to must also be set to

9600 baud. The same goes for the Wabi port assigned to the AIX tty. Unfortunately there is, no way to automatically check the consistency of all these configurations, so just remember to double check everything.

Once the device and AIX are configured, configuring the COM port under Wabi is a two-part process:

- Specifying COM port settings
- Connecting the COM port to an AIX serial device (native device name)

To configure the port settings, bring up the Wabi Configuration Manager and select the **Ports** option. The Port Settings dialog box (Figure 22) is then displayed. In this dialog box select a **COM** port and specify the serial connection information:

- Name of the port (COM1, COM2, COM3)
- Baud rate (110, 300, 600, 1200, 2400, 4800, 9600, 19200)
- Data bits, the number of bits used to represent each character (4, 5, 6, 7, 8)
- Parity (Even, Odd, Mark, Space, None)
- Stop bits, the number of bits sent to mark the end of each character, (1, 1.5, 2)
- Flow control (Xon/Xoff, None)

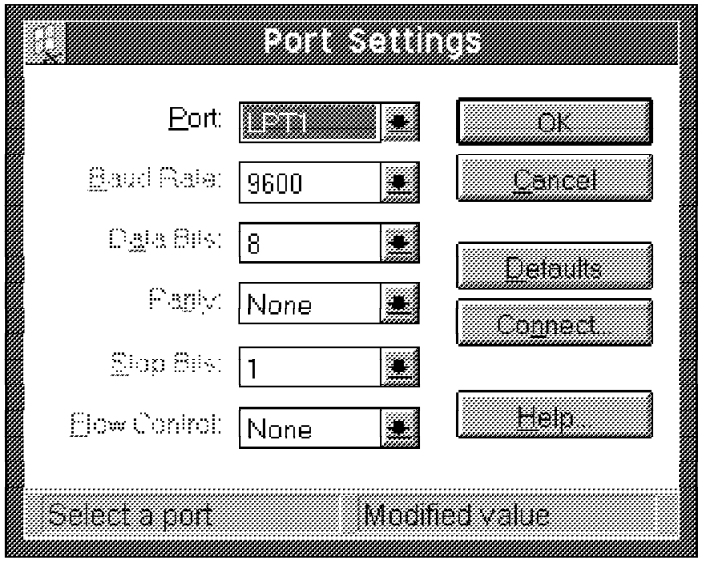

Figure 22. Port Settings Panel

Then, to connect the COM port to an AIX device, press the **Connect** button and the COM Port Connections dialog box appears(Figure 23 on page 48). This dialog allows you to specify which AIX serial devices Wabi should use when an application accesses the COM port.

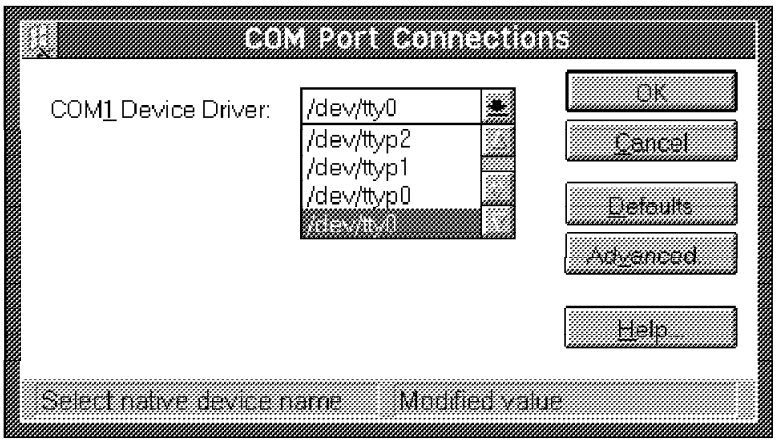

Figure 23. COM Port Connections Dialog Box

To assist you in selecting the AIX serial device, this dialog contains a drop-down COM Device list. This list is generated as a result of device search using a set of searching parameters. These searching parameters can be modified in the Advanced COM Port Options dialog box (Figure 24).

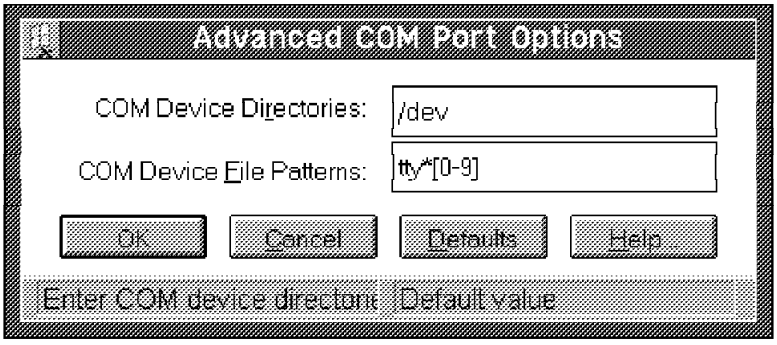

Figure 24. Advanced COM Port Options Dialog Box

To alter the device search, you can change both of the parameters in the Advanced Com Port Options dialog box:

- The COM Device Directory is a list of directories Wabi will search to locate the COM devices. You can include more than one directory by using commas to separate directory names. Do not include spaces.
- The COM Device File Pattern is a regular expression which defines to Wabi a pattern for the filenames of the tty drivers. You can also include more than one pattern separated by commas without any intermediate spaces.

If you specify the COM Device File Pattern with a * character, it will take a few minutes to bring up the COM Port Connections panel, as Wabi creates a list of all the files existing under the /dev directory.

The default COM device parameters and the search parameters are stored in the wabi.ini file, [IBM-RS/6000] section.

```
[IBM-RS/6000]
Devices.com1=/dev/ttya
Devices.com2=/dev/ttyb
Devices.com3=
Devices.com4=
Search.COMDevicePathPattern=/dev
Search.COMDeviceFilePattern=tty[a-z]
```

Note: In order to find the AIX tty devices, you must change the COM Device File Pattern option. The default COM Device File Pattern is set to tty[a-z] and doesn't match to the AIX tty devices. To correctly locate the AIX tty devices, you must change the COM device pattern to tty[0-9] or tty*[0-9]. This can be done by using the Advanced COM Port Options dialog box to modify the search pattern.

Any changes that you make to the COM device settings using the COM port dialog boxes will be stored in the [CommonSettings] section of the wabi.ini file. Wabi will first look in the [IBM-RS/6000] section and then update that information with the items found in the [CommonSettings] section. For example, the above mentioned change to cause the tty devices to be found under AIX will add the following settings to the [CommonSettings] section of your wabi.ini file and not the [IBM-RS/6000] section.

```
[CommonSettings]
Search.COMDeviceFilePattern=tty[0-9]
```

If you set COM1 to tty0 for example,using the COM Port Connection dialog box you will add this line in the [CommonSettings] section:

```
[CommonSettings]
Devices.com1=/dev/tty0
```

See Chapter 9, “wabi.ini File” on page 83 for more information about the wabi.ini file.

Chapter 7. X Windows Support

This chapter explains the way in which Wabi is supported on a graphical display using the X Window System. The discussion is separated into sections dealing with fonts, color and color maps, and the behavior of Wabi under X Window window managers.

7.1 Fonts

The handling of fonts is one of the most complex and confusing issues in Wabi. The font handling mechanisms are explained in the following sections.

7.1.1 Where Do Fonts Come From?

The fonts used by Wabi can be:

- The fonts available on the X server
- Windows specific Bitmap or TrueType fonts listed in the [FONTS] section of the win.ini file
- Fonts dynamically added by Windows applications during their execution

7.1.2 Fonts Supported by Wabi

Wabi supports:

- All ISO8859-1 fonts supported by the X server on which Wabi is running
- Microsoft Windows Bitmap fonts
- Microsoft Windows TrueType fonts

The Bitmap and TrueType fonts can be defined as followed:

Bitmap A Bitmap font is a collection of graphics images, one per character. These fonts tend to be designed for a particular size and resolution, and as a result do not scale very well to other sizes.

TrueType A TrueType font is a description of how to draw the outer edges of a series of characters. When using a TrueType font, Windows or Wabi calculates how to draw these hollow images for the desired size. As a result, TrueType fonts will appear correctly at any given scale, on almost any Windows-supported monitor and printer. Using TrueType fonts costs additional processing overhead. The TrueType font was one of the most significant advances in Windows 3.1 compared to Windows 3.0.

Wabi doesn't support:

- Adobe Type Manager Fonts

The best-know type-scaling program for Windows is probably Adobe Type Manager (ATM). ATM is not supported by Wabi because it is a 32-bit Windows program compiled with a Windows extender library. See 2.3, “Windows Application Program Interface” on page 8 for more information about the 32-bit interface.

Wabi 1.1 installs itself with three typeface families:

- Courier (courier bold, courier bold oblique, courier oblique)

- Helvetica (helvetica, helvetica bold, helvetica bold oblique, helvetica oblique)
- Times (times bold, times bold italic, times italic, times roman)

Windows 3.1 installs itself with five typeface families

- Times New Roman
- Arial
- Courier New
- Symbol
- Windgings

The fonts installed in your Windows configuration always appear in the [fonts] section of the win.ini file.

Before installing Windows, the [fonts] section of the win.ini file looks like as follows:

```
[fonts]
8514oem (8514/a res) = 8514oem.FON
```

After installing Windows, the [fonts] section of the win.ini file looks like as follows:

```
[fonts]
8514oem (8514/a res) = 8514oem.FON
Arial (TrueType)=ARIAL.FOT
Arial Bold (TrueType)=ARIALBD.FOT
Arial Bold Italic (TrueType)=ARIALBI.FOT
Arial Italic (TrueType)=ARIALI.FOT
Courier New (TrueType)=COUR.FOT
Courier New Bold (TrueType)=COURBD.FOT
Courier New Bold Italic (TrueType)=COURBI.FOT
Courier New Italic (TrueType)=COURI.FOT
Times New Roman (TrueType)=TIMES.FOT
Times New Roman Bold (TrueType)=TIMESBD.FOT
Times New Roman Bold Italic (TrueType)=TIMESBI.FOT
Times New Roman Italic (TrueType)=TIMESI.FOT
Symbol (TrueType)=SYMBOL.FOT
WingDings (TrueType)=WINGDING.FOT
```

When you install a new Windows application this section is updated according to the fonts available for both the display device and the printer device.

7.1.3 X11R5 Font Server

The X server code in X11R5 contains support for a network based font server. This Font Server is a program that runs on a host somewhere on the network and provides fonts to your X Server. This feature makes font administration much easier because the server manages its own font stores and provides its client X servers with fonts as requested. Also, this type of server architecture can provide several sources for a given font, which makes font access more reliable and less dependent on a single host.

Wabi will use the X Windows Font Server, if it is available, when it is building the font cache outlined in the next section.

7.1.4 Wabi's Font Cache

When Windows applications query Wabi for all available font data, Wabi needs to extract information from the X Window server. Since this query requires using the time consuming Xlib functions of XListFonts and or XLoadFont, this operation can consume an X server's attention for several seconds or even minutes at a time.

To avoid this delay, Wabi uses a program called wabifs to increase font performance. When Wabi is brought up it uses a default font cache or creates a new font cache by querying the available fonts from X Windows (including the font server if available). It then scales each font for use in Windows and then closes the font. This is the extent of X Window font interaction. Wabi then uses the wabifs program for all its font rendering needs.

Wabi maintains a series of default font caches for each of the different hardware platforms. The default font cache for IBM systems is named ibm.dflt.fc and is found in the /usr/lpp/Wabi/lib directory. When Wabi is first initialized on a particular X display, the font path of the display is compared to the font path stored in the default font path. If the paths match, the default version of the cache is loaded and used. This eliminates the need to build the cache. If the font paths differ, a new cache is built for the particular X display. During the cache building process, Wabi will display a dialog box similar to the one pictured in Figure 25 to notify you that the initialization will be delayed while the cache is being built.

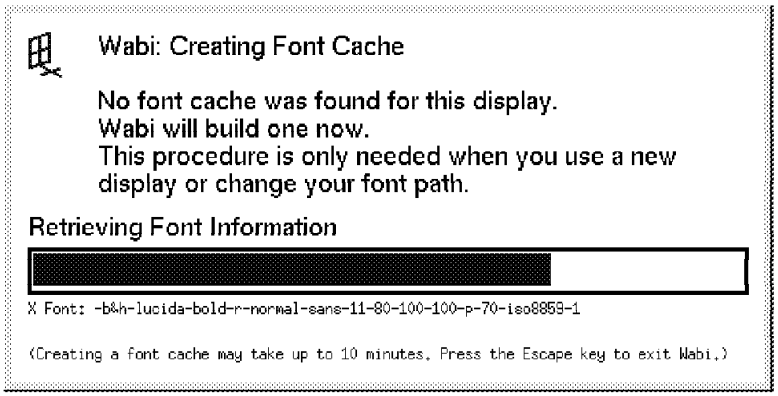

Figure 25. Font Cache Panel

Once the cache is created, it is stored in a file in the directory $HOME/wabi/fc. A separate font cache file is stored in this directory for each X display on which Wabi is run. Wabi uses the following naming convention for the font caches:

```
<display name><screen number>.fc
```

On subsequent Wabi restarts on a particular display, the font path of the current display is compared to the font path in the display's stored font cache. If the paths match, the saved version of the cache is loaded and used. This eliminates the need to rebuild the cache each time Wabi is started. If the font paths differ, a new cache is built for the display and stored for use on future restarts.

7.1.5 Font Cache on an Xstation

Wabi 1.1 will try and build the font cache every time it is started on an Xstation by different users. If the font path to the Xstation has not changed between Xstation sessions, this cache rebuilding process can be unnecessary overhead. You can eliminate this overhead by replacing the default font cache with a saved version of the font cache. Wabi will then load this saved version of the font cache instead of building a new cache each time. To do this:

1. Login as root (or su to root) on the server machine that is running the Xstation manager which is serving your Xstation.
2. Rename the default font cache from /usr/lpp/Wabi/lib/ibm.dflt.fc to /usr/lpp/Wabi/lib/ibm.dflt.fc.save.
3. Login on the Xstation using an ID other than root and start Wabi. This action will create a new font cache and put it in the ID's $HOME/wabi/fc/directory.
4. As the root user again, locate this new font cache file by finding the most recent cache file that contains your Xstation display name in the filename. Move this new cache file to /usr/lpp/Wabi/lib/ibm.dflt.fc.
5. Change the ownership of this cache file to be owned by user root and group system.
6. Change the permissions of this cache file to 644.

From now on, when anyone logs in on the Xstation it will use the prebuilt cache instead of building a new one each time. If the font path changes, you will need to repeat this procedure to build a new cache that contains the updated path.

7.2 Color and Color Maps

A color map in X Windows on AIX has 256 cells. Each one of these cells is capable of holding the definition for one particular color. This means that at most the color map can contain 256 unique colors. X Windows can support multiple color maps on the same X Server. However only one color map can be active at any given time. Suppose there are two different applications each using different color maps on the same server. When the user moves the mouse pointer from the windows of one application to the windows of the other application, X Windows will automatically switch the active color map to the map that is used by the application whose windows are underneath the pointer. This color map switching can cause two visual problems for the user:

1. The screen will appear to flash as the color maps are switched.
2. One of the applications can allocate a particular cell in their color map to be, for example, blue and the other application can allocate the same cell number in their own color map to be a different color such as red. When the X server loads the color map of the first application, all of the areas in its windows that were intended to be blue will be correctly displayed blue. However, since there can only be one active color map at a time, the windows of the other application that are still visible on the screen are now indexing into the wrong color map. Areas that were intended to be red are now displayed in blue. This indexing into an unknown color map can cause an applications windows to appear to be corrupted and unreadable.

To help minimize these problems X Windows maintains a default color map that all applications can share. When an application wants to display a unique (not already in the map) color, it can allocate one of the color cells in this default color map to hold the definition for the color. If the color map is full the X Server will return an error to the application. The application can at this point either switch to using a color already in the color map that is similar to the unique color it was looking for or it can create its own color map as above and allocate all the 256 cells for its own use.

The original concept behind Wabi was to allocate all the available color map cells and fill it with a selection of colors. This selection consists of several shades each of seven different colors(red, green, blue, cyan, magenta, yellow, and gray) and a color cube which is a set of colors that step through the color spectrum from black to white. The theory behind this approach was if an application asked for a new color it would be able to find a color in the color map that is reasonably close to the requested color. If all applications accepted this close color match the switching between color maps (flashing) would be minimized. However it was found later that, most applications were not written to accept this close match color and ended up creating their own color map. So now, instead of most applications sharing the default color map, most applications have their own color map and the flashing effect was increased.

Wabi's design was then changed to try and allocate the color cube in the default color map and then duplicate the default map for use as its own color map. In addition, seven new Wabi initialization variables have been created to control Wabi's use of the color maps. These new variables allow you to control how Wabi uses the default color map and the size of the color cube that Wabi will work with.

You can specify your desired settings for the new color variables in the win.ini file. Each variable can be set either in the [ColorMap] section or in the [display:0.0] section (for the X Server named "display" on screen 0) or both. This method allows some of the settings to be used only on some X Servers. For example, you may want to change settings depending on which X Server (24, 8 or 4-bit) you are using. Wabi sets the values for these seven variables using the following algorithm.

First, each variable is set to its default value. Next, the win.ini file is searched for an override value that may have been specified for each variable. Values found in the [display:0.0] section take precedence over those found in the [ColorMap] section. You may want to set your own default setting for a variable in the [ColorMap] section and then override your new default value in the [display:0.0] section of a particular display. Lastly, Wabi then compares the value set for each variable to determine if it is in an acceptable range of valid values. If the value is found to be out of range, the value is adjusted to the closest valid value.

Don't forget to use a DOS editor to modify the win.ini file since it is a DOS ASCII file. See 9.1, “How to Edit a DOS ASCII File” on page 83 for more information on editing this type of file.

The seven variables are listed in the following table.

Note: The last six variables are only significant if you are using an 8-bit PsuedoColor visual. A lot of display adapters use this PsudeoColor color technique in which each color map entry is a set of a specific red, green and blue intensities

that make up the desired color. Check your hardware documentation to determine if your display adapter is using an 8-bit PsuedoColor visual.

Table 2 (Page 1 of 2). Wabi Color Variables

Variable	Default Value	Legal Range	Description
UseRootWindow	See Desc.	0,1	This variable determines if it is okay for other code in Wabi to draw to and read from the root window. The default value is usually 1. The default value will be set to 0 if there are not the same number of entries in the default color map as in the Wabi color map. This variable should never need to be set by users. It was included for testing purposes, and for emergencies in the field.
PercentFree	50	0 - 100	This variable controls how many of the colors allocated in the default color map will be freed after copying to the Wabi color map. By default, half of the entries allocated will be freed.
Technicolor	See Desc.	0,1	This variable controls whether Wabi allocates colors from the default map and then copies them to the Wabi color map (0), or just allocates the Wabi color map as a 3/3/2 color map without modifying the default map (1). This variable defaults to 0 unless there is more than one hardware color map in the X Server for this screen. If there is more than one hardware color map, it is assumed one will be available for Wabi. Users on X Servers with more than one hardware color map may wish to try setting this to 0 if all hardware color maps are usually exhausted by other applications. Users on X Servers with only one hardware color map may set this to 1. In that case Wabi will go completely technicolor with respect to any other application running.
SolidColorCount	7	1 - 16	This variable defines how many shades of each of the seven colors(red, green, blue, cyan, magenta, yellow, and gray) are allocated. These solid colors are allocated first, before the color cube is allocated. A total of (7 * SolidColorCount) colors are allocated from the default color map using calls to XAllocColor. This variable is only used if Technicolor variable is set to 0.
RedCubeCount	5	4 - 9	This variable defines the dimension of the red component of the color cube. The color cube is defined by creating a cube that is RedCubeCount color shades by GreenCubeCount color shades by BlueCubeCount color shades. The number of colors actually allocated in the color table will in most cases be less than the size of the color cube plus (7 * SolidColorCount) because some of these entries may already exist in the color table and may be duplicates between cube and the solid color entries. This variable is only used if Technicolor variable is set to 0.

Table 2 (Page 2 of 2). Wabi Color Variables			
Variable	**Default Value**	**Legal Range**	**Description**
GreenCubeCount	5	4 - 9	Same as RedCubeCount except for green component of the cube.
BlueCubeCount	5	4 - 9	Same as RedCubeCount except for blue component of the cube.

The Wabi development team recommends the following procedure to help minimize the flashing effect when using Wabi with other X applications.

1. Set the PercentFree variable to 0.
2. Start Wabi before any other applications.

The combination of these two items minimizes the differences between the Wabi color map and the default color map which will minimize the flashing. However, the flashing effect can not be avoided if you are using color intensive applications which in most cases will use their own color map anyway.

7.3 X Window Window Managers

In terms of X Windows and common X Window window managers, Wabi is not a well behaved application. This is mainly due to the window-manager-unfriendly behavior of most Windows applications.

To explain these problems, we must first clarify some X Window terminology.

7.3.1 X Window Terminology

In a regular X Windows environment, applications work in coordinates relative to their own origin - the top left corner of their own window, and write text or graphics to their own window. The screen background that lies underneath all the application windows is known as the root window and provides an absolute or overall coordinate system for the screen. X Windows manages translating the application coordinates (relative to the origin of the application window) to root coordinates (relative to the origin of the root window) and determines where on the actual visible screen the text or graphics will appear, whether there is a window on top of the application that hides part of the application and so on.

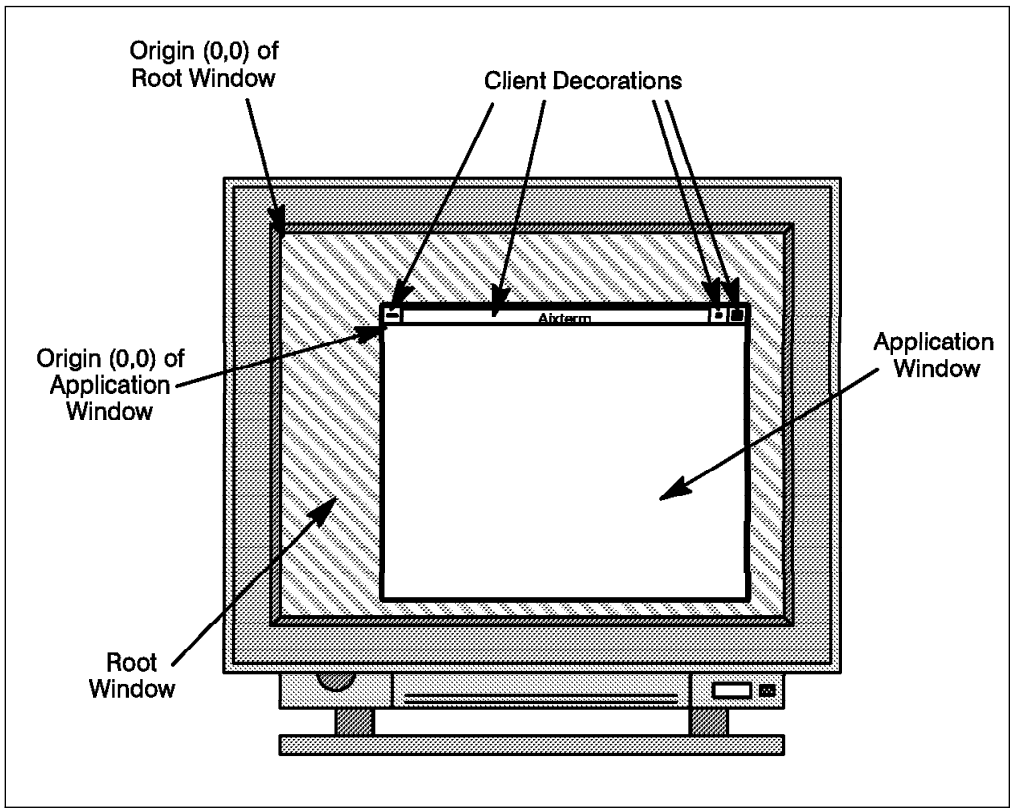

Figure 26. X Window Terminology

A window manager is an application that runs on top of X Windows and provides a consistent look and feel to the screen. It can provide a border around windows, or various client decorations such as the menu button, title bar and maximize and minimize buttons shown in Figure 26. The window manager also allows the user to move or re-size windows. The window manager provides the visual, user interface with a moving frame and window size as the user drags the mouse. Once the user releases the mouse button, the X server tells the application the new size of its window and the application can update the contents of its own window as appropriate. The window manager also has control over the position of icons, and determines which window is active and hence will receive keyboard and mouse input.

7.3.2 Behavior of Windows Applications

The following aspects of Windows applications made it difficult to integrate Wabi into X Windows environments:

- Windows applications rely on knowledge of their location in root coordinates (that is, relative to the origin of the root window). They often use this information to draw directly to the screen. If the window manager has moved the application's window, the text or graphics will appear in the wrong place. This also means that if another window has been placed on top of the Windows application, the text or graphics may appear on top of that window.
- Windows applications have control over the appearance and behavior of their own client decorations. To emulate this behavior the X Window window manager must surrender this control to the application.

- Windows applications rely on getting first access to any mouse or keyboard input. In an X Windows environment, the window manager first examines the input to see if it should act upon this input before passing the input to the application.
- Windows applications can still receive keyboard and mouse inputs while minimized or iconified. This is not permitted in a regular X Windows environment.

Wabi 1.1 solves these problems by by-passing the window manager for all windows created by Wabi. This means that the window manager does not control what happens to the Wabi windows.

This has several effects:

- Client decorations will be different for Wabi windows and other X Windows windows. Under most X Window window managers, it is possible to modify the colors, appearance and behavior of client decorations. These methods will not work with decorations on Wabi windows.
- It is possible under X Windows window managers to nominate the location of Icons. Wabi icons do not obey this rule, and may appear in a different corner to X Windows icons, or even on top of them.
- It is not possible to add extra options to the pull-down menus that appear when the user clicks on the menu button.
- Under most X Window window managers, it is possible to alter the Focus Policy. The focus policy is the method used to select the active window that will receive keyboard and mouse input. The most common focus policies are *explicit*, where you must click in a window to focus on or select that window, and *pointer*, where the focus moves to the window that the mouse is over. Wabi does not operate correctly with pointer focus policies, and it is recommended that you use explicit focus policy when using Wabi.
- When a Wabi-based window is maximized, Wabi will grab full control of the screen and will not allow you to use any other windows until you minimize the window.

These difficulties mean that the standard look and feel of the X Windows interface is to some degree broken by Wabi and Windows applications running under Wabi. However, generally this will be in the order of a minor annoyance rather than a serious problem preventing the correct or convenient use of Wabi or X Windows.

7.4 Wabi Desktop and Windows Integration

Using the command line options of Wabi you can run a Windows application from the AIX command line, either with or without invoking the Wabi shell (Application Manager or Program Manager) specified in the system.ini file. The command line options for Wabi are:

```
wabi [program] [-s program] [-display display_name] [parameters]
```

Where the options:

- -s, specifies that Wabi should run the program without running the Wabi shell specified in the system.ini file.
- -display, is used to display Wabi on a remote display device.

- Any parameters that you wish to supply to the Windows application can be specified at the end of the command. If you wish to pass the name of a file to the application, preface the file path name with r:. This allows Wabi to locate the named file off of the AIX root directory.

You can use these command line options to create icons or Motif menus to automate the starting of Wabi or of Windows applications. This can be done for either the AIX desktop or for a Motif Pop-up menu.

7.4.1 AIX Desktop

Wabi and Windows applications running under Wabi can be started from the AIX Desktop by defining the necessary desktop rule files and objects to call Wabi and Windows applications.

Below is an example procedure for creating a desktop icon that when selected will bring up Wabi. This icon will also accept a dropped icon from any .exe (executable) file that is to be started using Wabi. This example creates a desktop object called *Wabi* in your home directory. There are several ways to create objects under the desktop; this example uses a manual (file editing) approach because of the code complexity involved in defining the drop trigger that only accepts .exe files.

Note: The following procedure assumes that you have installed the Redbook Sample Tools and Utilities diskette that comes with this Redbook. See C.1, “Installing the Redbook Sample Tools and Utilities” on page 112 for more information on installing the Redbook tools and utilities.

1. Copy the Wabi.obj directory from the RedbookTools/XDesktop directory to your home directory. You can use the following command if you change your current working directory to the RedbookTools/XDesktop directory.

   ```
   cp -r Wabi.obj $HOME
   ```

2. Verify that the paths to the Wabi executable in the s1.objscr and d1.objscr files (located in your $HOME/Wabi.obj directory) match the installed location of Wabi. They should unless you have moved Wabi.
3. Start up the desktop (if it is not already up).
4. In your home directory you will find an icon named Wabi. When you select this icon Wabi should come up.
5. If you desire you can copy this object onto a desktop by dragging it out of your home directory and onto an open desktop. Figure 27 on page 61 is an example of what this would look like.

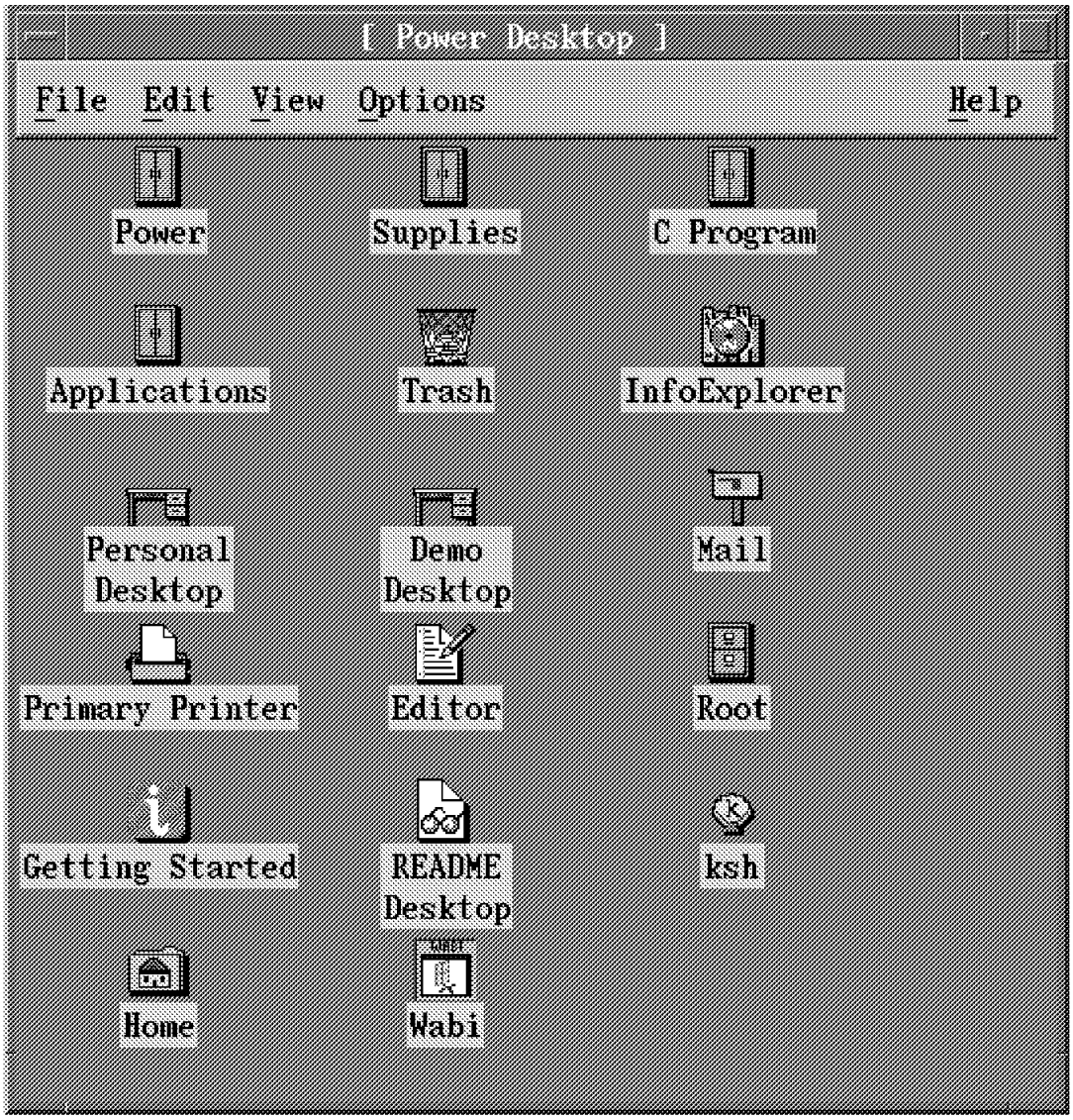

Figure 27. XDT Wabi Object on a Desktop

The desktop object that you copied from the RedbookTools/XDesktop directory contains the following items:

- colorl.px - This is the large color icon for the Wabi object.
- colors.px - This is the small color icon for the Wabi object.
- monol.px - This is the large monochrome icon for the Wabi object.
- monos.px - This is the small monochrome icon for the Wabi object.
- s1.objscr - This is an object script that defines the trigger action for double clicking on the Wabi object icon. The contents of this script can be found in C.9, “Sample Tool/Utility: Wabi.obj” on page 123.
- d1.objscr - This is an object script that defines the trigger action for dropping an icon on the Wabi object icon. The contents of this script can be found in C.9, “Sample Tool/Utility: Wabi.obj” on page 123.

Below is example procedure for creating desktop rules that will cause the desktop to recognize a Windows application and its data files. In this example, an icon_rule is created for the Microsoft Word application. The rule file will tell the desktop to do the following:

- Display a Microsoft Word icon when it encounters the winword.exe file.
- Allow the user to double click on this icon to bring up the Microsoft Word application.
- Display a Microsoft Word document icon when it encounters a file with the .doc extension.
- Allow the user to drag and drop the document icon onto the Microsoft Word icon. This action will start the Microsoft Word application and load in the dropped document.
- Allow the user to double click on a Microsoft Word document icon to have the Microsoft Word application started and the selected document loaded.

This procedure assumes that the Microsoft Word application is already installed and working under Wabi. To have the desktop do the above mentioned items, do the following:

1. Shut down the desktop (if it is already up).
2. If you have previously created a user rule file called .xdtuserinfo, add the contents of the xdtuserinfo_wabi file located in the RedbookTools/XDesktop directory to your user rule file. If do not have a user rule file, copy the xdtuserinfo_wabi file from the RedbookTools/Xdesktop directory to your home directory. You can use the following AIX command from inside the RedbookTools/XDesktop directory do this copy:

   ```
   cp -i xdtuserinfo_wabi $HOME/.xdtuserinfo
   ```

3. Verify the paths to the Wabi executable in the .xdtuserinfo file match the installed location of Wabi. They should unless you have moved Wabi.
4. Verify (change if necessary) the paths to the Word executable in the .xdtuserinfo file match the installed location of Microsoft Word.
5. Copy the Microsoft Word sample icons from the RedbookTools/XDesktop/wordicons directory into your user desktop icon directory. If you do not have such a directory, you can set one up with the following procedure:

   a. Create a directory in your home directory to hold the icons. This procedure assumes that you will call the directory bitmaps. Use the following AIX command from your home directory to create the directory:

      ```
      mkdir bitmaps
      ```

   b. Copy the contents of the sample wordicons directory into this new directory. From the sample directory you can use the following AIX command to do the copy:

      ```
      cp wordicons/* $HOME/bitmaps
      ```

   c. Add the path name of your new bitmaps directory to the xdt3.pictureDirectory desktop resource located in your .Xdefaults file. The following is an example of this resource statement:

```
xdt3.pictureDirectory: $HOME/bitmaps  \
 /usr/include/X11/bitmaps/xdt_p_small \
 /usr/include/X11/bitmaps/xdt_p_large \
 /usr/include/X11/bitmaps/xdt_i_small \
 /usr/include/X11/bitmaps/xdt_i_large
```

Note: If you are having trouble getting the desktop to display the new icons, try specifying the full path for your bitmaps directory. We did experience a problem using the $HOME variable in one test system environment.

6. Start up the desktop.

If you open the directory containing the Microsoft Word application, your directory should contain icons for both the application executable and any Microsoft Word documents. You can double click on the document icons to start up Microsoft Word and load the document. You can also drag a document icon onto the Microsoft Word application icon to start Microsoft Word and load the document. If you double click on the Microsoft Word application icon, Microsoft Word will start. You can also drag the Microsoft Word application icon and drop it on the Wabi icon created in the previous example to start Microsoft Word. Figure 28 is and example of a Microsoft Word directory containing these new icons.

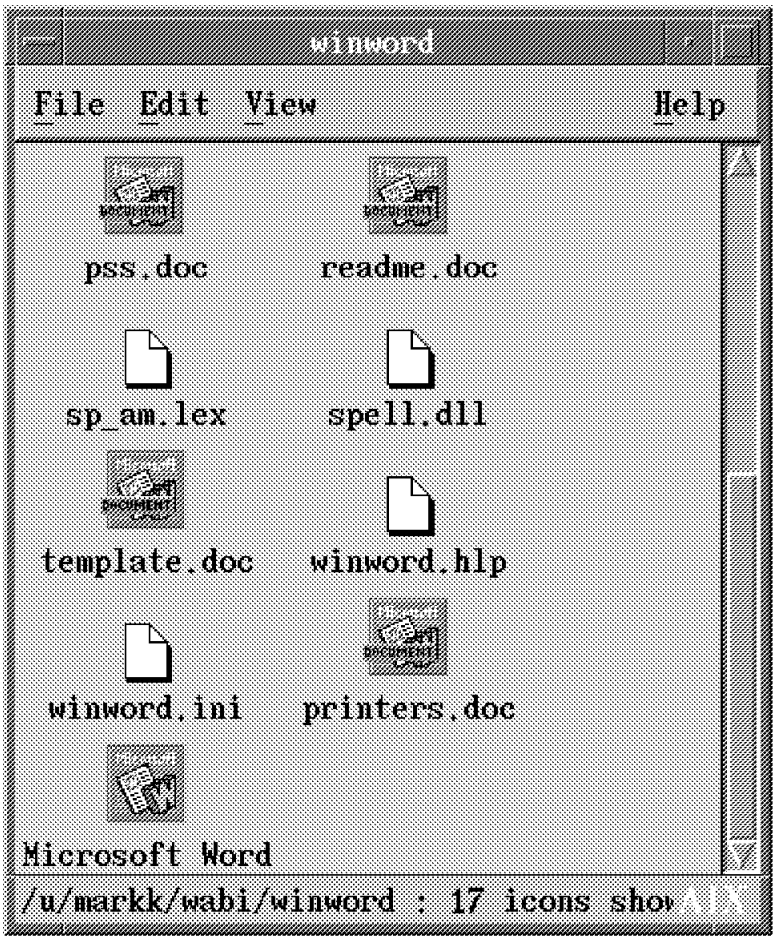

Figure 28. Microsoft Word Directory Integrated with Wabi

The contents of the sample rule file xdtuserinfo_wabi can be found in Appendix C, “Redbook Sample Tools and Utility Programs” on page 111.

7.4.2 Motif Menus

Wabi and Windows applications can be started through the Motif window manager menu system. The following procedure will define an entry on your Motif Root Menu that brings up a Wabi menu. On the Wabi menu we will define buttons for bringing up Wabi and the Microsoft Word Windows application.

1. Edit your .mwmrc file using your favorite editor. If you do not have a .mwmrc file located in your home directory, copy the system default file named /usr/lib/X11/system.mwmrc to your home directory and name it .mwmrc.

2. Locate the line that reads:

   ```
   Menu DefaultRootMenu
   ```

   This is the Root Menu definition. The items listed between the next { character and the matching } character make up the Root Menu definition.

3. Insert the following line into the list of Root Menu items anywhere below the line that contains f.title.

   ```
   "Wabi"      f.menu WabiMenu
   ```

   This line will cause Motif to add an option to the Root Menu that when selected will being up our Wabi menu.

4. Insert the following lines after the closing } of the Root Menu. Be sure to modify the paths to Wabi and the Word application if they are different from those listed below.

   ```
   Menu WabiMenu
   {
    "Wabi & Applications" f.title
    "Wabi"                f.exec "/usr/lpp/Wabi/bin/wabi &"
    no-label              f.separator
    "Microsoft Word"      f.exec "/usr/lpp/Wabi/bin/wabi \
                                  $HOME/wabi/winword/winword.exe &"
   }
   ```

   These lines define the Wabi menu titled "Wabi & Applications". This menu contains a Wabi button that will bring up Wabi. Below the Wabi button is a separator that separates the Wabi button from the Windows applications. Below the separator is a button for each (only one in this example) Windows application. Selecting the Microsoft Word button will use Wabi to bring up the Word application. Figure 29 on page 65 is an example of the default Motif Root Menu after it has been modified by this procedure.

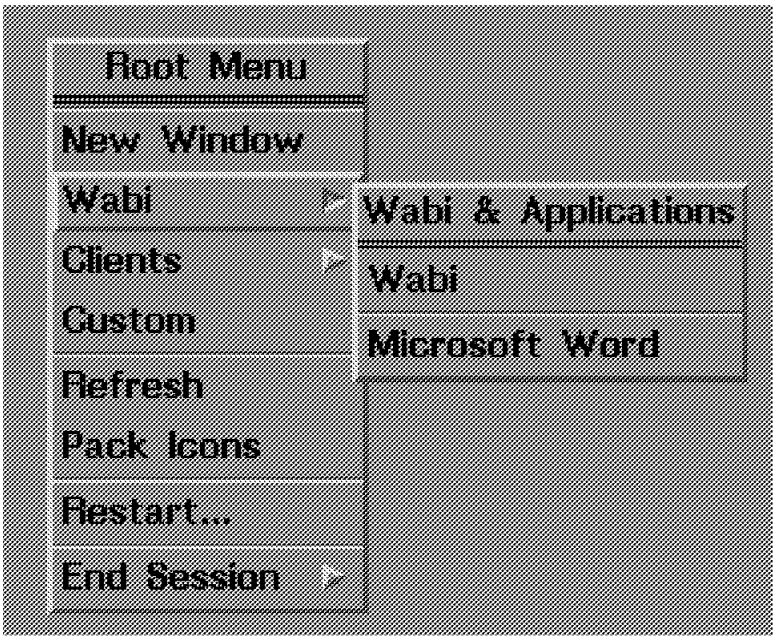

Figure 29. Motif Root Menu with Wabi

Chapter 8. Using Wabi in a Multiuser Environment

Microsoft Windows and its applications were designed to operate in a single user environment - one user on one computer. With the advent of networking, the picture changed only slightly. Although networked computers allow the sharing of data among users, we still have one person per computer. With the introduction of Wabi in the AIX multiuser environment, Windows and Windows applications are now being moved into an environment in which they were never intended to be used - multiple users on a single computer. This new environment for Windows introduces some new considerations that were never taken into account when Windows or its applications were designed. These considerations are:

- Since the products were not designed to be shared among a group of users, do we need to install one copy per user on the system? Or, is there a way that we can install only one copy, have the users share it, and thus save the wasted disk space of multiples version of the same product on the system?
- If we share a copy of the product will it work correctly? Does the product use any configuration files that are set up for the user and stored in the product directories? Remember, these products were designed to be installed and used by a single user. Modifying files in the product directory is not a problem if you only have one person using the product. In a multiuser shared product environment, this strategy will pose some very large problems, such as one user's configuration may not be desirable for another user.
- Does sharing a product pose any product licensing problems? Even if we could technically make a shared copy of the product work, we will need to be sure to have purchased enough licenses to support one per user. Or does getting a site license from the vendor make sense? Some products will support a networked installation. This type of license/product design is intended to allow you to install a server portion of the product on one machine and multiple client portions on other machines that share the server. This type of installation is usually licensed one per client machine.

This chapter will address using Wabi to run Windows and Windows applications in this new multiuser environment and will offer some insight into the above mentioned considerations by:

- Explaining how to set up Windows in a shared environment
- Explaining how to set up Windows applications in a shared environment
- Showing how to handle data files in a shared environment
- Giving an example on how to set up a shared environment

8.1 Sharing Microsoft Windows

Microsoft Windows 3.1 is an application that supports installation in a networked environment. Although this type of installation is intended to be used on a group of networked single user computers, we can make it work under Wabi on a single multiuser AIX system. This network installation procedure will allow us to save a considerable amount of disk space over installing a complete copy of Windows for each user. A regular complete installation is approximately 16MB per user. A network installation is approximately 16MB for a server, and 200KB for each user.

If you have even just a couple of users on a single system, the savings will be substantial.

In this networked installation approach, Windows is installed in two pieces:

- Client
- Server

The server portion(16MB) contains the bulk of the files needed to run Windows. In the Wabi/AIX environment, this portion can be set up to be shared among all users on a single system or on multiple systems by using a networked filesystem such as NFS or AFS. The client portion (200KB per user) contains just the configuration files that are unique to each user. Since these files are unique they are not intended to be shared among users.

Windows has an installation procedure for doing a network install. Unfortunately, it must be run from a DOS prompt. Since Wabi is not a DOS emulator, we cannot do the procedure exactly as stated by the Windows installation documentation. However with a few deviations from the installation instructions we can make it work in the Wabi/AIX environment.

Because the installation program must be run from DOS we have two alternatives for executing this program. We can create the client files and server files by installing Windows on a DOS Personal Computer and then copy the files from the PC to AIX. A better approach is to use a DOS emulator such as the AIX Personal Computer Simulator/6000. By using the PC Simulator the files can be placed by the installation program directly into the AIX filesystem and not have to be copied from a separate machine as in the DOS PC approach. In either case we will be running the Windows setup program with two separate options:

```
setup /A
setup /N
```

- The administrative option of setup (`setup /A`) transfers the files from the Windows diskettes to a network drive (you can use this command even if you don't have a network installed). This command will copy, expand, rename and mark as read-only the necessary server portion files from the Windows diskettes on to the hard drive.
- The `setup /N` command does the same for the client portion.

The following procedure will outline this process in more detail.

Remember to be sure and purchase enough licenses of Microsoft Windows to cover the number of users that you set up with the client portion to use the Windows product.

8.1.1 Procedure for Installing a Shareable Version of Microsoft Windows

This procedure will install a shareable version of Microsoft Windows for use with Wabi.

Note: The steps in this procedure assume you are using the E: drive. If you use a different drive, be sure to substitute your drive letter for E: in the rest of this procedure.

1. Create an AIX directory to hold the server portion of the Windows files. The location of this directory is up to you. This example will use a directory named /winshare. Be sure and set the permissions on the directory so that all the intended users will have read access.
2. Select a DOS environment either on a personal computer running DOS or with a DOS emulator such as the AIX Personal Computer Simulator/6000 to do the initialization install. If you elect to use the PC Simulator, you must start up the simulator with the `-E` option to tell the simulator to assign the E: drive to the AIX directory where the shared files will reside. This example would use `-E /winshare` as one of the PC Simulator flags. Also remember to use the `-case L` flag on the PC Simulator command to ensure that the filenames will be saved in the correct case for Wabi. See 5.3, “Using AIX Personal Computer Simulator/6000 as the DOS Emulator” on page 33 for more information about using the AIX Personal Computer Simulator/6000 under Wabi.
3. Insert the Windows Version 3.1 diskette labeled number one into the diskette drive.
4. At the DOS prompt, type the drive letter of the diskette drive (A: for example) to switch to the diskette drive.
5. Enter the command `setup /A`.
6. Follow the instructions on your screen. The setup program will prompt you to specify the network drive and directory into which you want to install the server portion of the Windows files. Enter `E:\WINDOWS` for this directory. You will also be prompted to specify group registration information (group name and company name). This information is stored and used when individual workstations are set up.
7. Create a new directory on the E: drive in which to install the client portion (`E:\WIN` for example).
8. Move to the directory where Windows is located (`E:\WINDOWS`).
9. At the DOS prompt, type `setup /N`. It is recommended that you use Express Setup; however, you can use Common Setup if you wish. The setup program will ask where it should put the client portion of the Windows files. Enter `E:\WIN`.
10. If you used the PC Simulator go to the next step. If not, copy from the personal computer the server portion from E:\WINDOWS into an AIX directory /winshare/windows. Be sure to copy the files names in lowercase on AIX.
11. Use the AIX `chmod` command to set read only access permissions to the files in the /winshare/windows directory.

    See 8.2.4.2, “File Permissions” on page 76 for more information about permissions. .

Perform the following steps for each user that will need to access Windows under Wabi.

12. Login as the desired user and start up Wabi.
13. Use the Drive Connection dialog box to assign a drive to the AIX directory containing the Windows files (`E -> /winshare`). It is recommended to use the same drive you used to create the server portion on your DOS personal computer. If you do not use the same drive letter, you will have icon problems.

See 4.4, “Moving Applications” on page 27 for more explanations about the icon settings.

14. Modify your $HOME/wabi/windows/progman.ini file with a Windows or DOS editor to introduce the new Windows groups (accessories, games and main):

```
[Settings]
display.drv=wabi.drv
Window=11 12 551 447 1
Order=1

[Groups]
Group1=C:\WINDOWS\WABI.GRP
Group2=C:\WINDOWS\ACCESSO0.GRP
Group3=C:\WINDOWS\GAMES0.GRP
Groupr4=C:\WINDOWS\MAIN0.GRP
```

15. Exit Wabi.
16. Modify the PATH in the $HOME/wabi/autoexec.bat file to include the following:

```
PATH=C:\WINDOWS;E:\WINDOWS
```

17. Copy the following files and ONLY these files from the client portion of the installation to your $HOME/wabi/windows directory. If you used the DOS PC method, copy the files from the PC to AIX. If you used the PC Simulator method, copy the files from the /winshare/win directory.
    - main0.grp
    - accesso0.grp
    - games0.grp
    - win.ini (First copy the old one to win.ini.old)

    After you finish copying these files you can either keep the /winshare/win directory around to install other users or you can delete this directory and create a backup copy of your $HOME/wabi/windows directory which can be used to set up other users.
18. Copy the following files from the server portion (/winshare/windows) into your $HOME/wabi/windows/system directory:
    - cpwin386.cpl
    - drivers.cpl
    - main.cpl
    - snd.cpl

    These files are used by the Microsoft Control Panel.
19. Also copy the following files from the server portion (/winshare/windows) into your $HOME/wabi/windows/system directory:
    - *.ttf

    These are font files and may be called in the win.ini file.
20. Restart Wabi.

8.2 Sharing Windows Applications

Applications will either come with a network installation option similar to the one mentioned for Windows or they are designed to be installed and used in a single system environment. Applications that are for single system environments are not easily migrated to a multiuser environment. This is largely due to the fact that the application may keep and maintain configuration files in the installation directory. If these files can be identified, they can sometimes be moved to a user home directory and with the proper pathing or links, can be set up to have a client/server like relationship as in networked applications. However, since this procedure can be very tricky and varies from application to application, it is not recommended. You may also find that the product's license agreement will not allow you to use the product in such a configuration. For these types of applications you are better off purchasing and installing one copy per user.

A lot of today's Windows applications do have network installation capabilities. This means that you should be able to install one part of the application as the server and the other part as the client. The client's part includes initialization (.ini) files that contain the preferences for the program's menu and selection tools. The bulk of the program is stored on a shared system where it's easy to access and maintain.

Unfortunately, this facility is not supported on Wabi 1.1. Later in this chapter we will discuss a possible workaround that might be used to overcome this limitation. But first, we will show you some sample results that we found when trying to use the network installation capabilities of a few applications. This information is included to illustrate the kinds of problems that you might run into when using applications that support the network installation capability.

8.2.1 Application Network Installation Problems under Wabi 1.1

The network installation feature has been unsuccessfully tested by the ITSO-Austin for the following three applications on Wabi 1.1:

- Microsoft Word
- Microsoft Excel
- AMIPRO

For Microsoft Word and Microsoft Excel, the problem that we encountered was that the setup program didn't recognize any Wabi drive as a network drive, and so it did not allow us to use the network installation option.

The problem was a little bit different for AMIPRO. We could install the files on the server using the network option but we had a problem when using the client setup. The client setup program didn't recognize the drive where the license file was copied (N:\AMIPRO\LETSHARE\AMIPRO.V30) and would not continue.

8.2.2 Using the Network Installation without Wabi Support

The following workaround overcomes some of the problems that we encountered while trying to use the network installation capabilities of some applications. This workaround has only been tested using the Microsoft Word application. Some of the concepts shown here might be applicable to the application that you are trying to install in a shared environment. The key message of this section is there are ways around some problems that you might encounter trying to use Wabi with

Windows applications in the AIX multiuser environment. You just need to keep an open mind when looking for alternatives.

As you may recall from the previous section, the problem that we encountered trying to do the network installation of the Microsoft Word application was that the installation program did not recognize a Wabi drive as being a network drive. Because of this Microsoft Word's setup program would not allow us to do a network installation.

To get around this problem we installed the server portion of Microsoft Word on a network of DOS personal computers first and then copied the server portion to AIX and Wabi. On a set of networked PC's we installed the server portion of the Microsoft Word application as instructed by the Microsoft Word installation instructions. We were very careful to make sure that the installation drive letters that we used were preserved between the PC and the Wabi environment. We installed ours in E:\winword. After the installation was completed on the PC, we copied the installed server directory tree from the PC to an AIX directory called /winshare/winword. We were careful to use lowercase names when we copied the files to AIX. This completed the server portion of the application. We were now ready to install the client portions under Wabi. First we deleted the Word installation on the PC's so we would not violate the application license agreement.

The rest of this procedure can be repeated as may times as you need to set up all the users for which you have purchased licenses of Microsoft Word. On AIX we brought up Wabi and assigned the shared /winshare directory (server portion) to the Wabi E: drive. See 6.2, “Disk Drives” on page 43 for information on accessing drives under Wabi. If you have followed the procedure outlined in 8.1.1, “Procedure for Installing a Shareable Version of Microsoft Windows” on page 68 you should already have this drive assigned. Notice that on the PC, the application was installed in E:\winword, and under Wabi the application is in the same place. We then ran the setup program again, this time under Wabi, and we used the copy that is located in E:\winword. We then followed the product setup instructions for a Workstation Installation (client). When setup asked us for a path to install the files we entered:

```
C:\winword
```

This action installed only the client portion in our home directory.

We now have the server portion on a sharable drive and the client part on our private drive. We then followed the rest of the applicable instructions for setting up the application in a networked environment.

8.2.3 Sharing Application Groups

You may want to display a consistent window to all users of Wabi. This window might include all programs that your company has site-licensed for each user. Notepad, File Manager, Microsoft Word and Microsoft Excel for instance, might belong on this window.

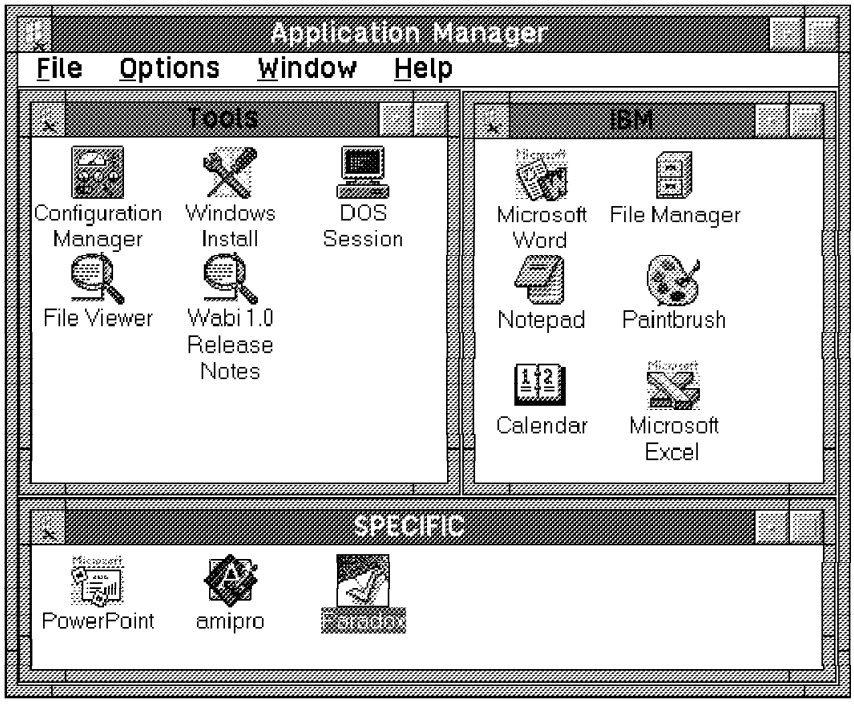

Figure 30. Application Manager Panel

One way to create this type of window is to define an Application Group that contains these icons (see the IBM application group in Figure 30). A second Application Group can contain icons for programs that are not licensed to all users, but only for certain individuals (see the SPECIFIC application group in Figure 30).

You must first create this group as the system administrator. This is necessary in order to create files with adequate permissions. See 8.2.4.2, “File Permissions” on page 76 for more information about file permissions.

To create an Application Group, you must:

1. Bring up the File pull-down menu from the Application Manager panel and select the **New** option from the menu to display the New Application panel (Figure 31).

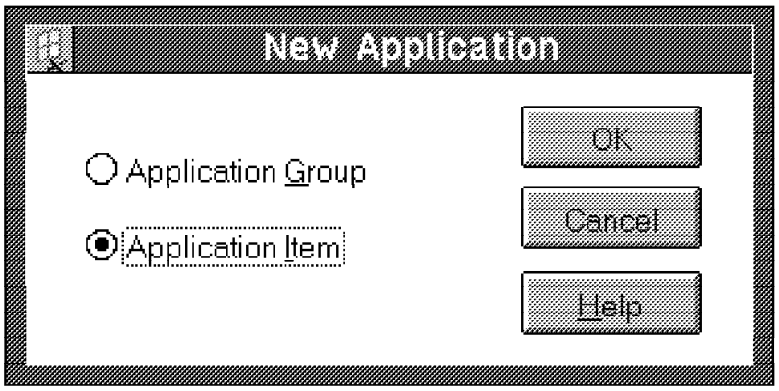

Figure 31. New Application Panel

2. Select the **Application Group** option in this panel. The Application Group parameters dialog box is then displayed. Enter the name of your company, for example, in the Identifier field of the dialog box. Then enter a full path name for the group file you are creating into the Group File field of the dialog box. For example, E:\groups\ibm.grp. Where E: is a shared Wabi drive connected to an AIX directory like the shared one (/winshare) that we have been using in this chapter. Figure 32 shows an example of this completed dialog box. Press the **OK** button when finished.

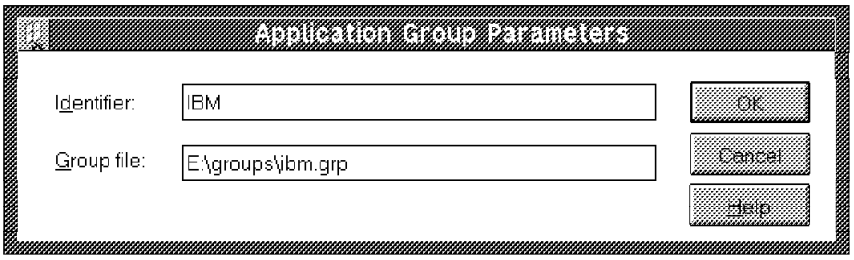

Figure 32. Application Group Parameters

3. Copy the the applications you want to display in this new group by using the drag and drop facility (shift button and left mouse button) on their icons. Save the configuration using the **Save Layout** option on the Options pull-down menu from the Application Manager panel.

These steps have created the ibm.grp file located in the /winshare/groups directory. Since /winshare is a shared directory only the system administrator should have read-write permissions on this new file. The other users will only have read permission.

The file progman.ini in your home/wabi/windows directory has been updated to include a new line in the [Groups] section:

```
[Groups]
Group<n>=E:\GROUPS\IBM.GRP
```

Where n is the group number that was assigned to this new group.

For each user that is to have the new group added to their window environment:

1. Be sure each user has access to these common applications and they have assigned the same Wabi drive letter to the shared application directory as you did.
2. Manually add the above group line into their [Groups] sections of their progman.ini file. If necessary modify the group number in the line to be the next sequential group number that is not already in the file. Be sure that a DOS or Windows editor such as Notepad is used to modify this file. An AIX editor will not put the correct line ending codes into the file.

The new group will now be available for each user that has modified their progman.ini file as above. Although the users will be able to change the location or the parameters of each icon in this new application group while Wabi is running, these changes can not be saved because E:\groups\ibm.grp is a read-only shared directory.

Note: Using your company name in the title of this menu, may remind the users that this is a company provided group and that it cannot be changed.

8.2.4 Controlling Access to Shared Applications

There are several different ways to control a user's access to applications in a shared environment. As the system administrator you can:

- Configure a user's Program Manager pull-down menus to eliminate the File option or just the Run option in the File pull-down menu.

  Note: This method will only stop inexperienced users from accessing these features.

- Restrict access to an application by using file permissions (Read Only, Hidden and others).

8.2.4.1 Controlling the Program Manager Pull-Down Menus

You can add commands to the progman.ini file to restrict some options on the Program Manager panel such as:

- Disabling the Run option from the File pull-down menu
- Disabling the Exit option from the File pull-down menu
- Disabling the Close option from the Program Manager pull-down menu
- Removing the File option from the Program Manager pull-down menu
- Disabling the user's ability to save the changes in icon locations when Wabi is exited, the user can still move icons around, but new positions are not restored in the new locations when Wabi is restarted

Note: Each user has their own copy of the progman.ini file. If you as the administrator change a user's file to implement these restrictions, the user can easily undo those changes and once again have access to the restricted functions. This procedure is not intended to be a security measure. It is only intended to help prevent inexperienced users from hurting themselves.

Using these restrictions will only work if a user is using the Microsoft Windows PROGMAN.EXE shell (see 4.5, “Changing the Windows Shell” on page 28 for more information). To implement these restrictions, you will need to add or modify the section called [restrictions] in the users progman.ini file.

Be sure to use a DOS or Windows text editor such as Notepad to make any changes to this or any other Windows configuration file. Other editors can corrupt these files. See 9.1, “How to Edit a DOS ASCII File” on page 83 for more information on editing these types of files.

You can add or change any of the following options in the [restrictions] section:

- NoRun=1 disables the Run option from the File pull-down menu
- NoClose=1 disables the Exit option from the File pull-down men
- NoFileMenu=1 disables the Close option from the Program Manager pull-down menu
- NoSaveSettings removes the File option from the Program Manager pull-down menu
- EditLevel=line to anything other than 0 disables user's ability to modify group windows, icons or icon properties (what program and parameters the icons run)
  - EditLevel=1 prevents deleting, creating, or changing group application names

- EditLevel=2 prevents deleting or creating program items
- EditLevel=3 prevents changing the Command Line of programs item
- EditLevel=4 prevents changing any properties of a program item

8.2.4.2 File Permissions

When looking at an AIX file under Wabi, the DOS file attributes are mapped or assigned as follows:

1. The userid that invoked Wabi is used to determine which of the AIX file permissions (user, group or other) to apply to the file.
2. If the user does not have write access to the AIX file, the DOS read-only attribute is set.
3. If the AIX file has the setuid bit set, the DOS hidden attribute is set.
4. If the AIX file starts with a dot character (.), and the filename is not . or .. then the DOS hidden attribute is set.
5. If the AIX directory attribute is set, the DOS directory attribute will be set.
6. The DOS archive attribute is set.

When a file is created under Wabi, the AIX. file permissions are set as follows:

1. The permission bits are first set to 666 (read and write for the user, group and others.
2. If the DOS read-only attribute is set, the AIX write bit is cleared for user, group and others.
3. If the DOS hidden attribute is set, the AIX setuid bit will be set.
4. The AIX file permissions will then be the least permissions allowed by the above and by the AIX umask.

Table 3 shows some sample mappings between DOS and AIX file attributes mappings.

Table 3. Example DOS and AIX File Attribute Mappings

DOS Attributes			**AIX**	
Read Only	**Hidden**	**Directory**	**Permissions**	**File Name**
X			-r--r--r--	myfile
			-rw-r--r--	myfile
		X	drw-r--r--	myfile
	X		-rw-------	.myfile

By knowing how this mapping scheme works, you can set up the appropriate permissions under AIX to restrict a user or a group of users access to a particular applications executable. For example, if you have an application (myapp) that you only want your department to access you can:

- Define the people in your department to be in their own AIX group (cathydept).
- Change owner's group on the executable to be the name of your department group.
- Remove read (r) permission from others.

This will limit access to the application to only those ID's defined in your department group. Executing the AIX command `ls -al` on the myapp.exe file would produce the following:

```
-r--r----- 24 cathy cathydept    123532  Mar 21 1993 myapp.exe
```

We have included in the Redbook Tools directory two sample shell scripts that can be used to help set up application permissions for a shared environment.

- give_attr: Will assign a given group name to an application directory/files. This script will also set the application to be read only so it can be used in a shared environment. For example:

  ```
  give_attr /winshare/myapp cathydept
  ```

  will set myapp and all of its files to be read only and assign them to the AIX group cathydept.

- add_to_grp: Will add a given user to a given group. For example:

  ```
  add_to_grp mark cathydept
  ```

  will add the userid mark to the AIX group cathydept.

For more information on these sample scripts see C.7, “Sample Tool/Utility: give_attr” on page 120 and C.8, “Sample Tool/Utility: add_to_grp” on page 122.

Note: The Properties dialog of the File Manager is not supported by Wabi 1.1. The values in this dialog do not necessarily reflect the attributes of the displayed file.

8.3 Sharing Data Files

Wabi allows you to share data files between multiple users as long as all the users can access these files under AIX. Assigning a drive letter to the directory containing the data files is the only action you have to do in the Wabi program. These files may be on a single system or on a remote system accessible via NFS or AFS.

Wabi also supports both file locking and file sharing:

- File locking prevents multiple users from accessing a file or a record at the same time.
- File sharing enables you to share files and data with others by controlling who can do what with a file at a given time.

8.3.1 File Locking and File Sharing

The file locking and sharing facilities of Wabi are advisory only. This means that only applications that are aware of, and respect file locks and file sharing will use these facilities. Other applications that do not support or respect these facilities could potentially overwrite or corrupt your data unless you use a different method of file protection. For example, suppose you have a data file that is shared between two applications that support and respect file locking. A third application that does not support file locking could open and modify the file, unless it was prevented from doing so by setting the appropriate file permissions. In this example, the two applications that are intended to access the data file could be set up to run from userids that are in their own AIX group. If the permissions on the data file were set to -rw-rw-r--, only those userids would be allowed to modify the file. Others would

be prevented from doing so by the file permission mechanism which all applications use.

File locking occurs automatically under Wabi. This facility is available to all supporting applications without having to explicitly set up anything in Wabi.

File sharing, on the other hand, must be enabled for each drive that contains files that you wish to have your supporting applications use. When you connect the AIX directory to a Wabi drive using the the Configuration Manager Drives option, select the File Sharing box to enable this facility for the drive. See 6.2, “Disk Drives” on page 43 for more information on how to connect AIX directories to Wabi drives.

Note: File sharing adds a significant amount of overhead to the Wabi process. You will notice a performance degradation on any drive that is using file sharing. For this reason it is recommended that you only use this facility when necessary and try to isolate those files that are to be shared on a Wabi drive of their own. This way the performance hit of file sharing will not effect any other files that are not using the facility.

8.4 Setting Up a Shared Environment (Example)

As you have seen in the previous sections of this chapter, there are a lot of things to consider when using Wabi in a multiuser environment. If you plan to use Wabi in a multiuser environment. It is best if you take the necessary time now to develop an implementation strategy that will support your multiuser needs not only today but in the future.

The following questions may help you to develop such a strategy:

- Windows
  - Will all users share the same copy of Windows, will each user have their own copy or a combination?
  - Do you need to support multiple native languages?
  - Do all users need to have access to Windows or just access to Windows applications?
- Windows Applications
  - Are you going to use shared applications, single user or both?
  - Do you need to use the Wabi file sharing facilities?
  - Will the shared applications/data files be stored on the same system or on multiple systems?
  - Do you need to restrict access to applications/data files?
- Miscellaneous
  - Who will install, upgrade and maintain the system and applications?
  - How many users will be supported?

Each of the answers to the above questions will effect the necessary setup for Wabi, Windows, Applications and the underlying AIX filesystem structure. Be sure to develop an implementation strategy that will accommodate all of your needs.

The following is an example implementation for a fictitious organization and its various needs.

Department XYZ has eight users. The mission of this department is to develop proposals for bringing in business to the company. The data processing needs of the eight users are as follows:

- Windows needs
  - All users will need access to Windows. This is so that they can have access to the File Manager and other Windows applets such as the calculator.
- Windows Application needs
  - Five of the users will need to have access to the Microsoft Word application so that they can jointly develop proposals. Occasionally one of these developers will need to be able to use the Microsoft Excel application to add cost information into the proposal.
  - Two of the users are accountants and will only need to have access to the Microsoft Excel spreadsheet program.
  - The remaining person is the project manager and will need to have access to both applications.
- Native Language
  - One of the proposal developers is from France and will need to have access to French & English versions of the Word and Excel applications. This person is the proposal developer who will need to incorporate the cost data.
  - The remaining seven department members will use English versions of their needed applications.
- Miscellaneous
  - The accountants and the project manager are on one system while the proposal developers and the rest of the company are on another system.
  - Cathy is the system administrator and the project manager. She will install and maintain all the software.

The following is an example implementation for the data processing needs for department XYZ.

Cathy (the system administrator) has purchased the following application licenses to support the users needs:

- Six licenses of the English version of Microsoft Word - (five developers + one project manager)
- One French license of Microsoft Word (one French developer)
- One French license of Microsoft Excel (one French developer)
- Three licenses of the English version of Microsoft Excel (two accountants + one project manager)
- Eight licenses of Microsoft Windows

Cathy has set up the applications to run in a shared environment by installing the applications on a shared disk. She has added the applications to an AIX directory, called /winshare, that was previously created by the company administrator with the ID compadm. This directory is used to hold all shared applications used by their company. Cathy and the accountants, being on a different machine than everyone else, will remotely mount this shared directory to their local filesystem using NFS. The Microsoft Windows was also previously installed by the company administrator, since other users in the company use this product. Cathy is the owner of the Word and Excel applications and her AIX ID is set up in the AIX groups word, excel, and staff. She set up the ID for the French developer to be in the french group. The remaining developers are in the word group and accountants are in the excel group. Cathy has also defined a project directory called /Ddrive that is to be used to hold common project related data files. This will also be remotely mounted for Cathy and the accountants.

The AIX directories are set up as follows:

Directory Name or File Name	Owner	Group	Directory Permissions
/winshare	compadm	staff	drwxrwxr-x
/winshare/windows	compadm	staff	drwxrwxr-x
/winshare/En_US	compadm	staff	drwxrwxr-x
/winshare/En_US/wordgb.20a	cathy	word	drwxr-x---
/winshare/En_US/excelgb.400	cathy	excel	drwxr-x---
/winshare/Fr_Fr	compadm	staff	drwxr-xr-x
/winshare/Fr_Fr/wordf_.20a	cathy	french	drwxr-x---
/winshare/Fr_Fr/excelf_400	cathy	french	drwxr-x---
/Ddrive	cathy	staff	drwxrwxrwx

Cathy has set up the Wabi drive connections as pictured in Figure 33 on page 81. The French developer will only have the M: drive connected, and not the L: drive. Everyone else will have the L: drive and not the M: drive. File sharing is only enabled on the D: drive since this is the only place where shared data files will be stored. Figure 34 on page 81 shows what the /winshare directory looks like. Figure 35 on page 82 is a side by side comparison of the English L: and the French M: drives.

This completes the setup for department XYZ.

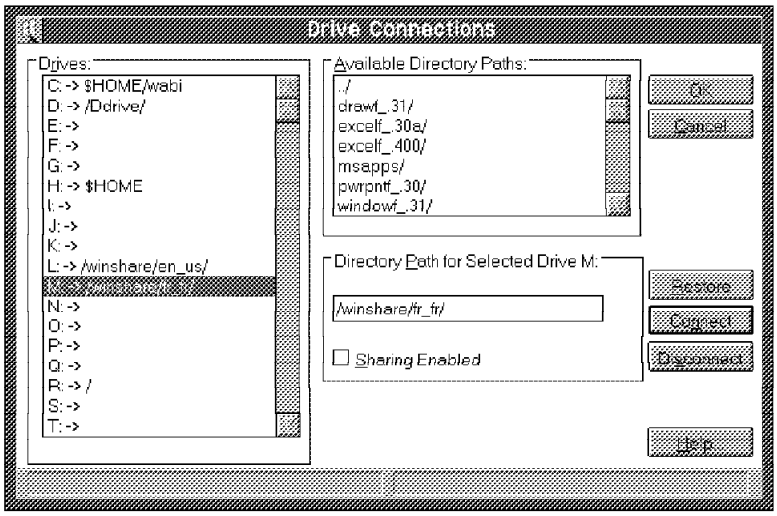

Figure 33. Example Filesystem Organization for Multiple NLS Applications. This picture shows the links between the AIX directories and the Windows drives (M: drive for French applications and L: drive for English applications).

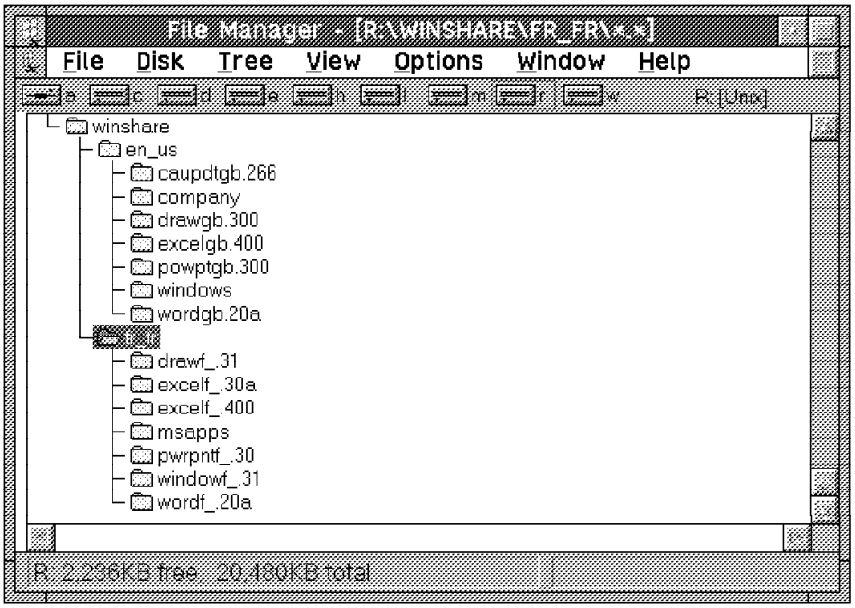

Figure 34. Winshare Directory Contents. This picture shows the AIX filesystem tree through the R: drive.

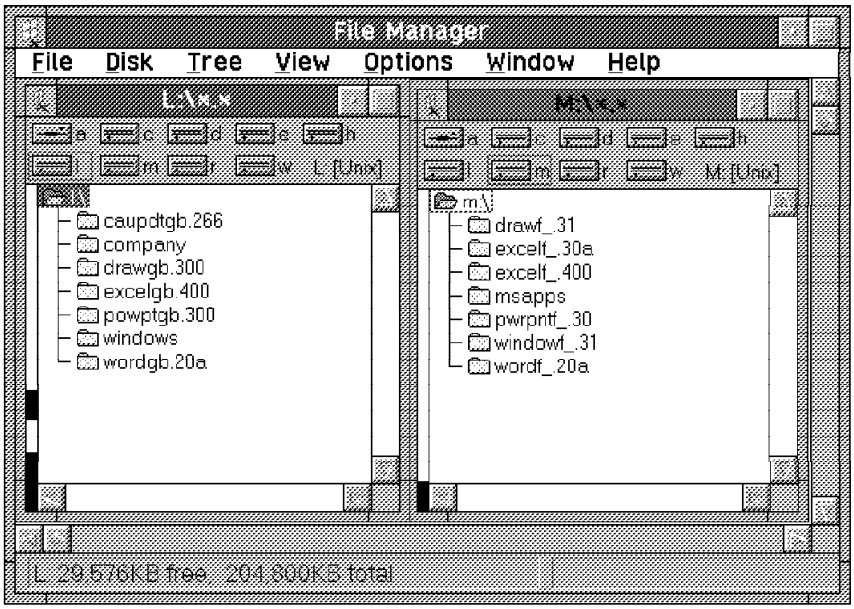

Figure 35. Multiple NLS Applications on a Single System. In this picture L: drive contains the English Windows shareable applications and M: drive the French versions.

Chapter 9. wabi.ini File

The Wabi initialization file (wabi.ini) contains information that defines your Wabi environment. If you change any Wabi configuration parameters they will be reflected in this file.

The wabi.ini file is located in your $HOME/wabi/windows directory.

In this chapter, we describe:

- How to edit a DOS file such as the wabi.ini file
- The format of the wabi.ini file
- How to change settings in the wabi.ini file

9.1 How to Edit a DOS ASCII File

The wabi.ini file is a DOS ASCII file and it must be edited with a DOS or Windows editor. DOS ASCII files are different from UNIX ASCII files. DOS uses the carriage return and the line feed characters to terminate a line. UNIX uses the line feed character only.

Due to this difference, if you want to edit a DOS ASCII file with an AIX editor such as vi, you must first convert it to a UNIX ASCII file. Then after you make your changes in the UNIX ASCII file, you must convert it back to a DOS ASCII file.

Wabi provides two tools which are located in your $HOME/wabi/wabihome/bin directory to do these conversions.

- unix2dos: converts ASCII files created with UNIX-based tools such as vi, to a DOS ASCII format. The syntax is:

  ```
  unix2dos UnixFileName DOSFileName.
  ```

- dos2unix: converts ASCII files created with DOS-based tools such as the Windows Notepad, to a UNIX ASCII format. The syntax is:

  ```
  dos2unix DOSFileName UnixFileName.
  ```

9.2 Format of the wabi.ini File

Wabi is implemented on several hardware platform. The wabi.ini file contains a section for each of the supported hardware platforms. For example, for the RISC System/6000, the section is titled [IBM-RS/6000]. The file also contains a section titled [CommonSettings]. This section is used for your own modification of the Wabi parameters. The items you place in this section have priority over items in any other section. This means, that if a particular parameter setting is found in both the [CommonSettings] section and in the [IBM-RS/6000] section the value found in the [CommonSettings] section will be used.

The wabi.ini file has the following sections that are of interest to AIX users:

- [IBM-RS/6000]
- [CommonSettings]

Some sections that are listed above may not appear in your wabi.ini file. For instance, you don't have a [CommonSettings] section if you didn't make any changes in the Wabi configuration parameters.

The sections and settings are listed in the wabi.ini file in the following format:

```
[section name]
keyname=value
```

Where `[section name]` is the name of a section. The enclosing brackets [ ] are required, and the left bracket must be in the leftmost column on the screen.

The `keyname=value` statement defines the value of each setting. A keyname is the name of a setting. It can consist of any combination of letters and digits in uppercase or lowercase, and it must be followed immediately by an equal sign (=). The value can be an integer, a string, or a quoted string, depending on the setting.

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

9.3 Changing Settings in the wabi.ini File

When you install Wabi, a wabi.ini file is created with default values assigned to the various settings.

Wabi uses the setting of the appropriate machine to set the default parameters in each of the Configuration Manager options (Ports, Printers, Drives, Diskettes, Mouse, Sound and DOS) dialog boxes.

There are two ways to change the wabi.ini settings:

- You can use the Configuration Manager options to change the settings. This is the safest and recommended approach because there is no need to manually edit the wabi.ini file.
- You can use a text editor, such as Windows Notepad, to edit the wabi.ini file directly. This approach is not recommended unless you are very familiar with the file and its contents. Incorrect parameters in this file can cause undesirable results. Remember, to *never* use a formatting editor like Microsoft Word or an AIX editor on this file. These editors will corrupt your wabi.ini file.

After you have made any changes to the wabi.ini file, you must restart Wabi in order for the new settings to take effect.

Caution

Always back up your wabi.ini file before you make changes so that you can restore the original file in case you accidentally damage the wabi.ini file or make changes that cause problems when running Wabi.

Here is the original [IBM-RS/6000] section of the wabi.ini file(Wabi 1.1):

```
[Version]
Config=1.1

[IBM-RS/6000]
Devices.disketteA=/dev/fd0
Devices.disketteB=
Devices.com1=/dev/ttya
Devices.com2=/dev/ttyb
Devices.com3=
Devices.com4=
Drives.C=$HOME/wabi
Drives.D=
Drives.E=$PWD
Drives.F=
Drives.G=
Drives.H=$HOME
Drives.I=
Drives.J=
Drives.K=
Drives.L=
Drives.M=
Drives.N=
Drives.O=
Drives.P=
Drives.Q=
Drives.R=/
Drives.S=
Drives.T=
Drives.U=
Drives.V=
Drives.W=$WABIHOME
Drives.X=
Drives.Y=
Drives.Z=
DriveFlags.C=Fixed
DriveFlags.D=Fixed
DriveFlags.E=Fixed
DriveFlags.F=Fixed
DriveFlags.G=Fixed
DriveFlags.H=Fixed
DriveFlags.I=Fixed
DriveFlags.J=Fixed
DriveFlags.K=Fixed
DriveFlags.L=Fixed
DriveFlags.M=Fixed
DriveFlags.N=Fixed
DriveFlags.O=Fixed
DriveFlags.P=Fixed
DriveFlags.Q=Fixed
DriveFlags.R=Fixed
DriveFlags.S=Fixed
DriveFlags.T=Fixed
DriveFlags.U=Fixed
DriveFlags.V=Fixed
DriveFlags.W=Fixed
DriveFlags.X=Fixed
DriveFlags.Y=Fixed
DriveFlags.Z=Fixed
```

```
PC_Emulator.Command=
Printers.name_lpt1=<Default Printer>
Printers.name_lpt2=<Default Printer>
Printers.name_lpt3=<Default Printer>
Printers.command_lpt1=/bin/qprt -B an -T%t -P%p
Printers.command_lpt2=/bin/qprt -B an -T%t -P%p
Printers.command_lpt3=/bin/qprt -B an -T%t -P%p
Search.COMDevicePathPattern=/dev
Search.COMDeviceFilePattern=tty[a-z]
Search.FDDevicePathPattern=/dev
Search.FDDeviceFilePattern=fd[0-9]
```

Chapter 10. Performance

The benchmark tests described in this chapter were performed with the goal of exploring the following questions:

- How much effect does the model of RISC System/6000 used have on performance?
- How much effect does the graphics adapter used have on performance?
- How does system memory effect Wabi response time?

It is important to note that these results have not been subjected to any formal or informal IBM reviews. They were performed using pre-release software and the results may bear no relation to results that would be obtained with official release code. These results are provided to allow IBM staff and customers to roughly compare the performance within different models of the RISC System/6000 and to provide a starting point for your own testing or prototyping. These results should not be compared with Windows applications running under Wabi or native Microsoft Windows 3.1 on other architectures.

10.1 Processor Model and Graphics Adapter Testing Procedures

This series of tests was intended to compare the performance using different models of RISC System/6000 and graphics adapters. In these tests, we attempted to maintain all other factors as constant as possible. They were performed on systems configured as workstations with a single test user active.

All RISC System/6000 models tested were configured with 32 Megabytes (MB) of system memory. We determined by watching vmstat during practice runs that no paging was occurring and thus memory was not a factor during this testing. Paging space was set at 64 MB. The systems tested all used a common, single 1 GB disk for the operating system and all data. No network devices were used.

The tests were performed using AIX Version 3.2.5 and AIXwindows Version 1.2.5. The fix U426451 was also applied to fix the refresh problems on some graphics adapters. In order to reduce external influences wherever possible, TCP/IP was disabled for all testing. Testing was performed with Wabi running under X Windows and the Motif window manager. No other X Windows clients were running. The X Window screen saver was turned off as there were noticeable differences in benchmark performance when the screen was not being displayed. After starting each test, the keyboard and mouse were not touched until the completion of the test.

10.1.1 WinTach 1.0 Benchmark

The Wintach** benchmark was created by Texas Instruments in 1992 to allow comparison of different graphics adapters for PC systems. It runs under Microsoft Windows 3.1 and consists of four discrete tests that simulate the graphics demands of Word Processing, CAD, Spreadsheet and Paint applications. Each benchmark is run by clicking on an icon which starts the particular application and follows a fixed set of procedures common to users of that application, such as changing fonts, moving around the document, and modifying justification for the word processing

benchmark. The performance in each test is rated, relative to a 386DX-20 PC using a standard VGA graphics adapter. When all four tests have been performed, an Overall WinTach RPM rating is calculated by averaging the four individual figures. The Overall WinTach RPM is again rated relative to the a 386DX-20 PC using a standard VGA graphics adapter.

It is possible to select the graphics resolution that will be used for the WinTach benchmarks. This has a considerable bearing on the results obtained. We performed all benchmarks at the maximum resolution of 1280x1024 pixels. We performed each section of the benchmark three times to ensure that the results were remaining consistent. In all cases, the variation in the results on the three runs was less than five percent - in most cases, it was less than two percent. The Overall WinTach RPM rating is based on the last test performed in each application. A larger WinTach rating indicates a better performance.

The WinTach benchmark is a good measure of the graphics performance of a system. It could be used as a guideline for a customer that will be performing graphics intensive work such as Computer Aided Design (CAD) or Desktop Publishing.

10.1.2 Dhrystone 2.0 Benchmark

The Dhrystone benchmark is a small C program that performs some very basic integer instructions. The program used is based on version 2.0 of the Dhrystone benchmark as implemented in Microsoft QuickWin**.

The program used allows the user to specify the number of loops through the code to be executed. It measures the time taken to execute these loops then calculates from this a rating for the number of loops that can be performed per second. Thus a larger number indicates higher performance. For our measurements, we used 500,000 loops, taking around nine minutes on the RISC System/6000 model 520. We ran the benchmark three times, and based the result on the average of the last two runs in order to minimize the effects of software caching. In all cases, the variation between runs was less than 1 percent.

The Dhrystone benchmark is small enough that the entire benchmark code will probably fit into the instruction cache on most modern workstations. Thus the Dhrystone benchmark measures only the raw integer computation speed of the processor without taking into account input/output performance, memory bandwidth or other factors. It is used here to illustrate the difference between the different RISC System/6000 models, and to compare with the Excel Recalculation Benchmark below, which does include the effects of these additional factors. The Dhrystone benchmark does not provide a good indication of any real-life application.

10.1.3 Excel Recalculation Benchmark

This benchmark is based on a full application - Microsoft Excel 4.0a. We used a spreadsheet with 104 columns, each containing 200 random 6 decimal place numbers totalled at the bottom of each row. We timed the recalculation of this spreadsheet, recording the results in a second spreadsheet. We used a macro to perform the tests and timing. The macro first loaded both spreadsheets, then wrote a timestamp to the second spreadsheet. It then fully recalculated the main spreadsheet ten times before writing a second timestamp and calculating the elapsed time.

We found that this benchmark was also very sensitive to what was displayed on the screen. If the entire spreadsheet was displayed and had to be updated, the benchmark was up to nine percent slower than only displaying a small segment. Since we already had the WinTach benchmark measuring the display speed, we ran the benchmark with Excel maximized (taking up the complete display), but with the main spreadsheet window sized down to display only three rows and ten columns of the spreadsheet. The second (results) spreadsheet and the macro sheet were also shown on the screen. We ran the macro three times, and based the result on an average of the second and third runs to minimize the effect of software caching. The value given for this benchmark shows the time taken for ten recalculations, and hence a smaller value indicates a better result in this benchmark.

The Excel Recalculation Benchmark provides a good real-life test of processor speed and should provide a good indication of relative performance in computing intensive applications.

10.1.4 Results

Table 4. Wabi Performance on Various Processor Models and Graphics Adapters

Model	Adapter	WinTach Overall RPM	Dhrystone	Excel Recalc (Min:Sec)
520	CDGA	13.33	941.60	12:50
	GT3	7.95	941.60	12:59
	GT3i	7.98	942.50	12:56
	8Bit 3D	6.01	941.60	12:59
	24Bit 3D	6.01	941.60	12:54
	GT4x 24Bit	8.12	939.80	12:53
350	CGDA	21.37	1953.10	6:30
	GT3i	15.10	1945.50	6:26
	8Bit 3D	8.42	1956.50	6:29
	GT4x 24Bit	15.68	1960.80	6:13
220	GT1	11.56	1031.00	15:06
	GT3i	6.23	1057.10	15:08
250	GTX150	58.23	3681.40	4:20
	GT3i	25.23	3610.15	4:21

10.2 System Memory vs. Wabi Response Time on a Workstation

Tests of Wabi performing the Excel benchmark test on a sample hardware configuration while varying only the amount of available system memory between tests, revealed the following:

- Response times steadily decreased as memory was added until system memory reached 16 MB.
- System memory amounts greater than 16 MB had negligible impact on Wabi response time.

Chapter 11. Wabi Around the World

Wabi 1.1 has support for international keyboards but currently only supports the English language. Wabi uses several environment variables to determine its international settings.

11.1 Environment Variables

Wabi uses the following environment variables:

- The LANG variable allows you to specify the language used and the keyboard type. Remember, only the English language is supported in Wabi 1.1.
- The WABI_KEYB variable overrides the keyboard specified by the LANG variable and allows you to use a different keyboard than the national default one.
- The WABIDIR variable can be used to override the default WABIDIR environment variable setting of $HOME/wabi. This feature will allow you to have multiple Wabi environments for the same user. For example, one English environment and one French environment.

Since Wabi 1.1 only supports one language (United States English), it's not necessary to specify the LANG environment variable. The message catalogs and the English libraries reside only in the /usr/lpp/Wabi/lib/locale/en_us/wabi directory. The Wabi messages are only used in the pop-up menus of the application groups. If you install an NLS version of Microsoft Windows, and if you change the shell to the Microsoft Windows shell (PROGMAN.EXE), the pop-up menus of the Application Manager will be displayed in the correct language. See 4.5, “Changing the Windows Shell” on page 28 for more information.

Wabi 1.1 does support native language keyboard specification. Table 5 on page 92 lists valid values for the WABI_KEYB variable.

Table 5. Supported International Keyboards	
WABI_KEYB Variable	**Language or Country**
da	Denmark
de	German
de-ch	German - Switzerland
en_uk	English - United Kingdom
en_us	English - United States
es	Spain
es_la	Latin America
fi	Finland
fr_be	Belgium
fr_ca	French - Canadian
fr_fr	French - France
fr_ch	French - Switzerland
it	Italy
nl	Netherland
no	NorWay
pt	Portugal
sv	Sweden

If you wish to have multiple NLS versions of Windows and applications, you can set up Wabi to switch between the environments by changing the WABIDIR environment variable. If you plan to do this, you need to copy your $HOME/wabi directory to create multiple Wabi environments. For example, you might have $HOME/wabi_us for the English version of Windows and applications, and $HOME/wabi_fr for the French versions. You can then use the WABIDIR variable to tell Wabi which version to initiate.

In the Redbook Tools directory we have included a sample shell script that can be used to automatically switch between Wabi environments by interrogating the system LANG environment variable. See C.3, “Sample Tool/Utility: go_wabi” on page 114 for more information. The shell script looks like this:

```
#!/bin/ksh
#############################################################################
###                                                                       ###
###  Project:                   Wabi Redbook Tools                        ###
###  Module:                       go_wabi                                ###
###  Authors :          Cameron Ferstat and Catherine Chevallier          ###
###  Date:                           17 Jan 94                            ###
###-----------------------------------------------------------------------###
###  Description: Wabi utility shell script                               ###
###                                                                       ###
###               The shell script switches the WABI directory according ###
###               to the LANG variable environment and then runs Wabi.    ###
###                                                                       ###
#############################################################################
```

```
if [ $LANG = En_US ] ; then WABIDIR=$HOME/wabi_us ;export WABIDIR; fi
if [ $LANG = Fr_FR ] ; then WABIDIR=$HOME/wabi_fr ;export WABIDIR; fi

/usr/lpp/Wabi/bin/wabi
```

To use this shell script, you must:

1. Set the language variable to the appropriate language, such as:

   ```
   export LANG=Fr_FR
   ```

2. Run the go_wabi shell script.

This script assumes that you have set up your various Wabi environments in your $HOME directory to be named: wabi_xx. Where xx is an identifier for your desired language. In this script, xx is either us or fr for United States English or French.

Figure 36 shows a French Wabi environment. The French Program Manager Panel supplied with the French Version of Microsoft Windows has been set up. All the Windows applets are in French. Three French Windows applications have been installed (Excel, WinWord and PowerPoint). Since Wabi 1.1 only supports the English language, the Wabi tools remains in English.

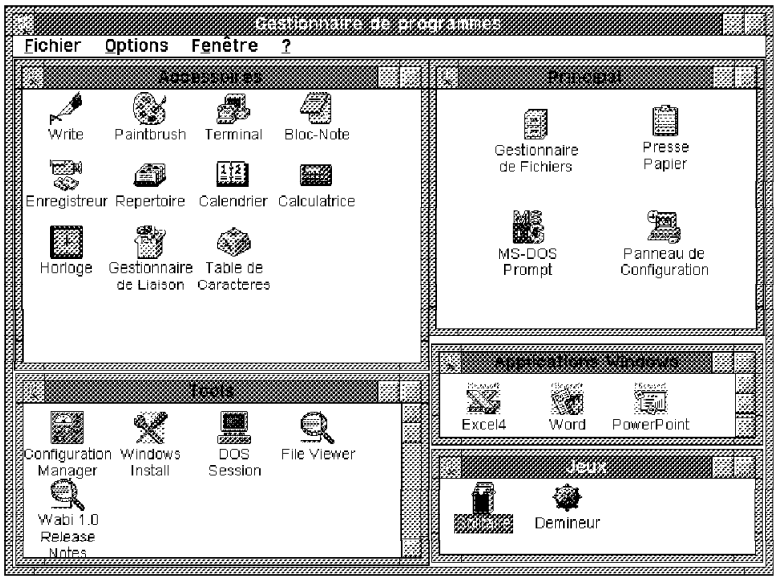

Figure 36. Wabi Environment for the French Version

The following figures show two French Windows applications:

- Figure 37 on page 94 is a Windows applet: the File Manager (Gestionnaire de Fichiers)
- Figure 38 on page 94 is a Windows application: WinWord

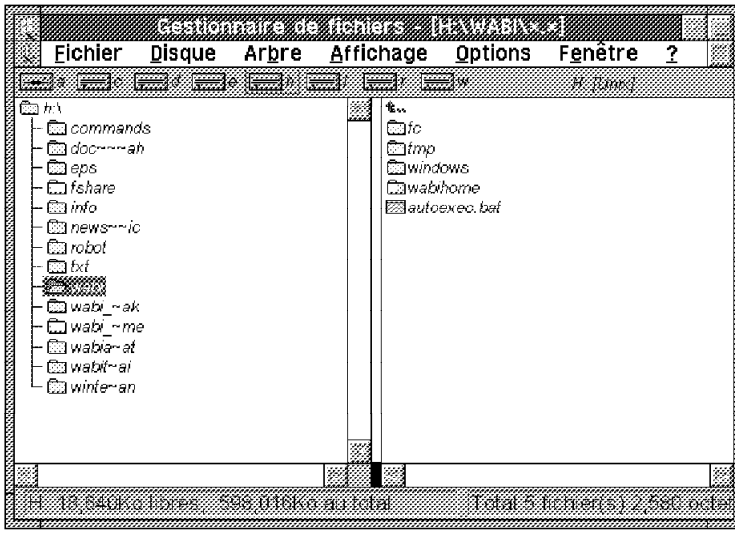

Figure 37. French File Manager Supplied with French Version of Microsoft Windows

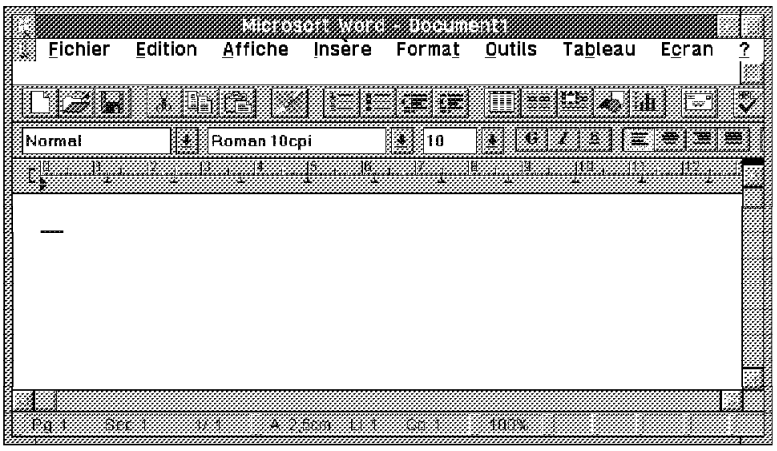

Figure 38. French WinWord Launched under Wabi

Chapter 12. Frequently Asked Questions

This is a collection of answers to questions that are frequently asked by Wabi users. Much of the information in this chapter has already been given elsewhere in this guide, however it is presented again in this format for ease of access.

12.1 What Does Wabi Stand for?

Wabi is not an acronym. It is simply a name and is a trademark of Sun Microsystems Inc. Wabi should always be written with a capital *W* and lowercase *abi*. See also 1.2, “What Does Wabi Stand For?” on page 2.

12.2 What Windows Applications Can I Run?

The following applications are qualified by IBM as being compatible with Wabi 1.1. This means they have been tested and all functions operate in the same way as when running under native Windows:

- Aldus PageMaker 4.0
- Borland Paradox for Windows (requires Microsoft Windows 3.1 to be installed)
- Borland Quattro Pro for Windows
- CorelDRAW 3.0
- Harvard Graphics for Windows
- Lotus 1-2-3 for Windows 1.1
- Lotus AmiPro 3.0
- Microsoft Excel 4.0
- Microsoft PowerPoint 3.0 (requires Microsoft Windows 3.1 to be installed)
- Microsoft Project 3.0
- Microsoft Word for Windows 2.0
- PROCOMM PLUS for Windows 1.0
- WordPerfect for Windows 5.2

Other applications that conform to Microsoft Windows 3.1 application program interface (Win16 API) conventions may operate correctly under Wabi 1.1. Lack of inclusion in this list does not mean that the application will not work - only that it has not been fully tested.

For information on some of our experiences with non-certified applications, see 4.3, “Non-Qualified Applications” on page 26.

12.3 What Version of AIX Do I Need?

Wabi 1.1 is supported by IBM on all RISC System/6000 models running AIX/6000 Version 3.2.5 and beyond.

To determine the level of your AIX operating system, type the command:

```
$ lslpp -L bos.obj
 Description                                          State     Fix Id
 ---------------------------------------------------  ------    ------------
 bos.obj 3.2.0.0
   3250 bos Maintenance Level                            C     U491123
   3250 AIX Maintenance Level                            A     U493250
     HFT Configuration Utilities                         C     U423532
     Kernel                                              C     U423980
     Device Drivers                                      C     U423981
```

Look at the number before the AIX Maintenance Level. If the number is 3250 you are on AIX level 3.2.5. The system shown above is running on level 3.2.5 with three additional updates beyond that level (for the HFT, Kernel and Device Drivers).

If the AIX Maintenance Level shows number 3240, you are using AIX Version 3.2.4. You must upgrade your system before installing or running Wabi.

If you get the following response:

```
$ lslpp -L bos.obj
lslpp: illegal option -- L
Usage: lslpp {-A|-d|-f|-h|-i|-l|-p} [-B | -I] [-acJq] [-O{[r][s][u]}]
             [product ... | fix_id ... | all]
```

you are on an AIX level below 3.2.4 and must upgrade your system before you can run Wabi.

12.4 What Version of X Windows or AIXwindows Do I Need?

Wabi 1.1 is supported by IBM on AIXwindows Environment/6000 1.2.5. This corresponds with Release 5 of the X11 X Window system (X11R5).

To determine the level of AIXwindows that your system is running, execute the following command: (Note the capital X in X11rte.obj)

```
 Description                                          State     Fix Id
 ---------------------------------------------------  ------    ------------
 X11rte.obj 1.2.3.0
   3250 X11rte X11-R5 Maintenance Level                  A     U491119
     AIXwindows Run Time Environment                     A     U409194
     AIXwindows Run Time Environment                     A     U411705
```

In the listing above, the level of AIXwindows appears as 1.2.3. Version 1.2.5 is a set of fixes on top of version 1.2.3. If the system has the 3250 X11rte X11-R5 Maintenance Level applied as shown above, then the system is at the correct level.

12.5 Do I Need to Install Microsoft Windows 3.1?

Wabi 1.1 includes native re-implementations of six Microsoft Windows 3.1 DLLs. Most other Windows DLLs will run under Wabi using the 386 instruction emulator. These DLLs are not provided with Wabi but can be installed as part of Microsoft Windows 3.1 or be included with a particular application. For a list of the natively re-implemented and non-native DLLs, see 2.2, “More Detail” on page 6.

If the application that you wish to run requires only the DLLs that are included with Wabi, you will not need to install Microsoft Windows 3.1 to run these applications. If your application uses the non-native DLLs, and does not provide these DLLs, you will need to install Microsoft Windows 3.1 or another application that includes the DLLs.

Of the 13 applications that are qualified for Wabi 1.1, only two applications require Microsoft Windows 3.1 to be installed. They are:

- Microsoft Powerpoint
- Paradox

Microsoft Windows 3.1 also includes several applets (small applications) including Write, Paintbrush, Terminal, Calculator and Games. These applets are not included with Wabi 1.1. If you require these applets, you must install Microsoft Windows 3.1.

The final reason for requiring Microsoft Windows 3.1 is to obtain the Windows help functions. The program WINHELP.EXE is not delivered as part of Wabi. To obtain WINHELP.EXE you must install Microsoft Windows 3.1.

12.6 How Many Microsoft Windows 3.1 Licenses Do I Need?

Microsoft Windows 3.1 running under Wabi is licensed in exactly the same way as when running on a PC. You must have a license for each user that will install Windows under Wabi. These can be individual licenses or part of a corporate or site license. For more information, contact your Microsoft Windows 3.1 dealer.

Note that it may not be necessary to install Windows to use Windows applications. For more information, see 12.5, “Do I Need to Install Microsoft Windows 3.1?”

12.7 What Level of WinAPI Does Wabi Support?

Wabi 1.1 supports the Win16 16-bit Application Programming Interface (API), as implemented by Microsoft Windows 3.1. It does not support the Win32 API used by the Windows NT operating system. For more information on WinAPI levels, see 2.3, “Windows Application Program Interface” on page 8.

12.8 Does Wabi Support 32-Bit Applications?

In order to support 32-bit applications, Wabi would have to support the Win32 32-bit Application Program Interface (API) or at least a subset of this API. Wabi 1.1 does not currently have this support. For more information on WinAPI levels, see 2.3, “Windows Application Program Interface” on page 8.

12.9 Can I Develop Windows Software on Wabi?

Wabi provides a binary compatibility environment for programs that conform to the Windows Application Program Interface (WinAPI) definitions, but there is currently no interface provided to allow programmers to develop Windows API programs under Wabi 1.1.

12.10 Does Wabi Support WinSock?

Wabi 1.1 does not support the WinSock sockets interface. This means that most Windows networking software will not operate under Wabi 1.1.

12.11 Does Wabi Support Dynamic Data Exchange (DDE)?

The library used for DDE, ddeml.dll, is not implemented natively by Wabi 1.1 and is not included as part of the files shipped by Wabi. However, ddeml.dll will execute under the 386 instruction emulator, so if you install Microsoft Windows 3.1 or an application that includes the ddeml.dll library, then applications that use DDE should function normally.

12.12 Does Wabi Support Object Linking and Embedding (OLE)?

The two OLE libraries, olecli.dll and olesrv.dll, are not implemented natively by Wabi 1.1 and are not supplied with Wabi. They will execute under the 386 instruction emulator. If you obtain the DLLs by installing Microsoft Windows 3.1 or an application that includes the DLLs, applications that use OLE should operate correctly.

12.13 Does Wabi Support Windows Enhanced Mode Operation?

Yes - Wabi's instruction emulator is based on the 386 instruction set, allowing Wabi to support Windows enhanced mode operations. In fact, Wabi always operates in enhanced mode and does not support standard mode operation. For more information on Windows modes, see 2.5, “Windows Modes” on page 10.

12.14 What about Memory Management?

All Windows applications running under a single instance of Wabi share a single memory address space to allow the correct operation of Windows applications that use shared memory as a method of inter-process communication. The down side of this implementation is that if a Windows application crashes, you will probably have to stop and restart your entire Wabi session.

For more information on the way Wabi handles memory, see 2.4, “Memory” on page 9.

12.14.1 Extended Memory

Extended memory is supported by Wabi. Applications that require extended memory will operate correctly.

12.14.2 Expanded Memory

Microsoft Windows 3.1 and Windows applications do not normally use expanded memory. Thus expanded memory support is not required under Wabi. For information on using DOS programs that use expanded memory under Wabi, see Chapter 5, “DOS Emulation” on page 31 and AIX Personal Computer Simulator/6000 documentation.

12.15 What Are All These Files With ~ in the Name?

DOS filenames are limited to an eight character name and a three character extension, separated by a dot (.) character. UNIX filenames are much more flexible than this. In order to allow Wabi and your Windows applications to access UNIX files with names that do not fit the eight character prefix and 3 character suffix restriction of DOS filenames, it is necessary to translate the filenames into legal DOS names.

The Wabi routines that perform this mapping are explained in detail in 2.7, “Filename Translation” on page 12. The tilde character (~) is used by the routines to signify that the filename has been mapped, and to pad shorter filenames to the required length.

12.16 Why Do My Files Have a ^M at the End of Each Line?

UNIX and DOS use different characters for marking the end of each line in ASCII text files. If you edit a DOS file in a UNIX text editor such as vi, you will see one of these characters, the Carriage Return character (ASCII 13) represented by a Control M or ^M.

You can use the utility programs *unix2dos* and *dos2unix* to convert a text file between DOS format and UNIX format. For more information see 4.6, “Moving Files between Wabi and AIX” on page 29.

12.17 Does Wabi Support Adobe Type Manager Fonts?

Wabi 1.1 does not support the Adobe Type Manager or Adobe Type Manager fonts. For more information on Wabi font support, see 7.1, “Fonts” on page 51.

12.18 Can I Run Two Copies of Wabi on My Screen?

It is only possible to run a single copy of Wabi on any particular X Window display. This is due to the way in which Windows and Windows applications are written. If two copies were to be run, it would not be possible to determine which Windows application should have the keyboard and mouse inputs.

For more information on the behavior of Wabi in a Windows environment, see 7.3, “X Window Window Managers” on page 57.

12.19 How Can I Cheat on Minesweeper?

Actually we haven't heard this question asked very frequently however it is fun to demonstrate to the type of customers that tend to read Windows bulletin boards and would be likely know that this method exists with native Windows. We often use it as light relief when using Minesweeper as an example of Wabi's ability to run programs from a UNIX command prompt without starting the Application Manager.

With the Minesweeper window active, type `xyzzy`.

Now hold the Ctrl button and move the mouse pointer around the Minesweeper window. Watch the pixel in the very top left hand corner of the screen. You will see this pixel flash black and white. When the pixel is black, there is a mine under your mouse pointer.

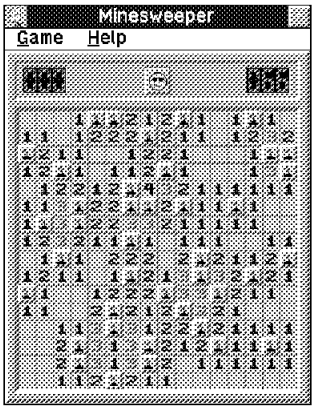

Figure 39. Minesweeper

Chapter 13. Common Problems

This section collects some of the most common problems that are encountered by Wabi users and attempts to answer them. Much of the information in this chapter has already been given elsewhere in the book, however it is presented again in this format for ease of access.

13.1 Cannot Load VER.DLL

If you see the warning dialog box shown in Figure 40, it is possible that you are trying to install Microsoft Windows 3.1 by running setup from the A: drive. Do not install Microsoft Windows 3.1 in this manner. Instead, double-click on the **Install Windows** icon in the Wabi Tools group.

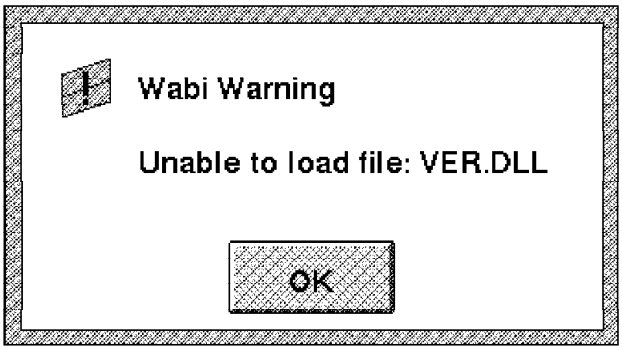

Figure 40. Warning Dialog

13.2 Couldn't Exec WINHELP.EXE

The warning dialog shown in Figure 41 is most commonly caused by attempting to use the Windows help functions without installing Microsoft Windows 3.1. The program WINHELP.EXE is not supplied with Wabi for copyright reasons, and must be supplied by installing Microsoft Windows 3.1.

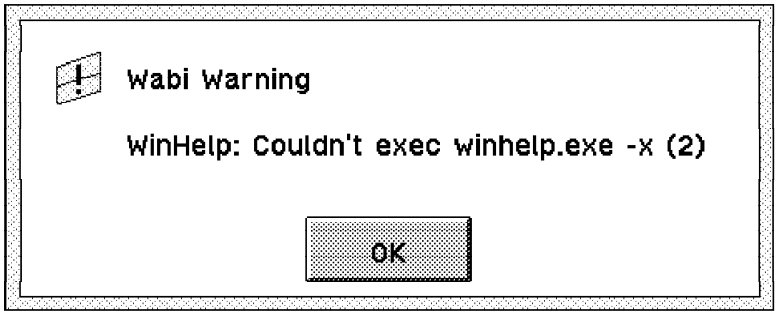

Figure 41. WinHelp Warning Dialog

13.3 Screen Doesn't Refresh Properly

There is a known problem that occurs when using Wabi on GT3, GT4e, GT4, GT4x, GT4i and GT4xi graphics adapters. The problem involves windows that are exposed (by moving another window that was obscuring part of the window) not being correctly refreshed with their contents.

A fix for this problem is available from your local AIX Software Support Center. For AIXwindows Version 1.2.5 (X11R5), the required fix is PTF U424296.

To determine if you already have the appropriate PTF installed, execute the command:

```
$ lslpp -hB U424296
```

If the required fix is installed, the output will appear similar to:

```
  Name
  --------------------
    Fix Id  Release         Status     Action     Date       Time     User Name
    ------- --------------- ---------- ---------- ---------- -------- ----------
  bth: /usr/lib/objrepos
  X11rte.obj
    U424296 01.02.0003.0000 COMPLETE   APPLY      02/15/94   11:59:54 root

Path: /etc/objrepos
  X11rte.obj
    U424296 01.02.0003.0000 COMPLETE   APPLY      02/15/94   12:00:23 root
```

If the fix is not installed, output will be similar to:

```
lslpp:  There is no fix id in /usr/lib/objrepos, /etc/objrepos,
        or /usr/share/lib/objrepos that matches "U424296".
```

If required, order the PTF from your local AIX Software Support Center.

13.4 X Station Cursor is a Red Block

There is a known problem with the AIX Xstation Manager/6000* Version 1.4.1 which causes the Wabi cursor to appear as a red block. This generally only occurs on Model 140 and 150 Xstations.

The fix for this problem is to apply Program Temporary Fix (PTF) U422610, which should be available from your local AIX Software Support Center. To determine if you have this PTF installed already, type the following command:

```
$ lslpp -hB U422610
```

If the required fix is installed, the output will appear similar to:

```
  Name
  --------------------
    Fix Id  Release         Status     Action     Date       Time     User Name
    ------- --------------- ---------- ---------- ---------- -------- ----------

Path: /usr/lib/objrepos
  x_st_mgr.obj
    U422610 01.04.0000.0000 COMPLETE   APPLY      11/02/93   14:25:52 root
```

```
Path: /etc/objrepos
  x_st_mgr.obj
    U422610 01.04.0000.0000 COMPLETE   APPLY      11/02/93   14:28:11 root
```

If the fix is not installed, output will be similar to:

```
lslpp:  There is no fix id in /usr/lib/objrepos, /etc/objrepos,
     or /usr/share/lib/objrepos that matches "U422610".
```

13.5 Symbol XtStrings in sh Is Undefined

If you receive the following errors when starting Wabi:

```
$ wabi
Creating /home/user/wabi
Could not load program /usr/lpp/Wabi/bin/wabiprog
Symbol XtStrings in sh is undefined
Symbol XtShellStrings in sh is undefined
Error was: Exec format error
```

you are most probably trying to run Wabi on a system that is running release 4 of the X Window System Relase (X11R4). Ensure that you are running AIXwindows 1.2.5 which uses X11R5. For information on how to check the level of your AIXwindows software, see 3.2, “Wabi Prerequisites” on page 18.

13.6 0K Free Space with AFS

Many people have complained that Wabi does not recognize available disk space on AFS filesystems. The filesystems will show under the file manager as 0K used and 0K free. This does not cause problems when reading files from the disk - only when installing programs. The problem is only seen if the installation program actually attempts to check for sufficient disk space. Microsoft Excel is an example of a program that does this.

One workaround for this problem is to temporarily move your Wabi drive off AFS. This could be performed using the following steps:

```
$ cd $HOME ; pax -rw -pe wabi /tmp
$ mv wabi wabi.bak                 (Just in case)
$ ln -s /tmp/wabi $HOME/wabi
Install Excel
$ rm $HOME/wabi
$ cd /tmp ; pax -rw -pe wabi $HOME
Start Wabi and check everything is OK
When you are satisfied everything is OK, then
$ rm $HOME/wabi.bak
```

13.7 Window Manager Problems - Focus, Icons, and Window Visibility

You are quite likely to experience problems with the control of your Wabi windows and icons in relation to other X Window windows and icons. These problems could include:

- Sometimes you cannot move other windows in front of Wabi windows.

- You cannot change focus to other windows when Wabi is showing error windows.
- You cannot control the placement of Wabi icons.
- Wabi icons appear on top of X Window icons.

These problems are caused mainly by the way in which Microsoft Windows 3.1 Windows applications are written, and the way that Wabi was implemented to allow for this. For more information on these problems, see 7.3, “X Window Window Managers” on page 57.

13.8 Going Technicolor

A problem common to most Wabi users is when the Wabi window has focus (is the active window on the screen), the colors on all other windows change. If you then change focus to one of the other windows by clicking on it, its colors will be correct but Wabi's colors will be wrong. This problem is often referred to as "going technicolor" and is common in X Window environments.

The problem arises because most X Servers can only display colors from a single color map at one time. The color map that is used for the entire screen will be the color map of the window that has focus at that time. Wabi allocates a separate color map of its own and allocates colors in that color map. Wabi copies some entries from the existing color map that is shared by most other applications, then allocates the rest of the color map according to its own requirements. Applications use an index to point to a color in the color map. If Wabi's private color map is active, that index may point to a totally different color.

There are several settings that can be used to change Wabi's use of the color map. These settings are described in 7.2, “Color and Color Maps” on page 54.

13.9 Cannot Access Diskette Drive from Outside Wabi

There is a known problem that relates to the way Wabi handles the diskette drive. If you have accessed the diskette drive from within Wabi by either of the following methods:

- Selecting the A: drive under the file manager.
- Opening a document from the A: drive under a Windows application.

Wabi will then hold the diskette drive open and will not release the drive for other UNIX applications such as tar or backup until you completely exit from Wabi.

If you encounter this problem you will see errors such as:

```
tar: /dev/rfd0: A device is already mounted or cannot be unmounted.
or
restore: 0511-158 Cannot open /dev/rfd0, A device is already mounted or cannot
be unmounted.
```

The problem occurs because Wabi was originally developed to be run on Sun Workstations. On Sun workstations, the diskette drive is mounted as a filesystem. AIX handles the diskette drive in a different manner. A workaround was written to allow AIX access to the diskette drive allowing normal operations for all Windows applications running under Wabi, however this workaround had the restriction that

once the diskette drive is opened by Wabi, it will not be closed until Wabi exits. To get around this restriction would have required large modifications to Wabi that were not possible to include in Wabi 1.1. This problem should be corrected in later versions of Wabi.

This problem can also prevent you from starting AIX Personal Computer Simulator/6000 since at startup time, PC Simulator looks at the diskette drive to see if there is a diskette that it should boot from. If PC Simulator cannot access the diskette drive, it will not run.

13.10 Cannot Start AIX Personal Computer Simulator/6000 Under Wabi

One reason for PC Simulator failing to run under Wabi is caused by the way Wabi locks the diskette drive after it has been accessed by a Windows application. When PC Simulator is started, it looks at the diskette drive to see if there is a diskette that it should boot from. If Wabi has the diskette drive locked, PC Simulator will fail to start.

The only workaround to this problem with Wabi 1.1 is to start PC Simulator before accessing the diskette drive under Wabi. If you have already accessed the diskette drive, you will have to exit and restart Wabi.

Appendix A. Wabi File Layout

There are two main areas where Wabi files are placed. The Wabi executables, utility programs, natively implemented libraries and other files common to all Wabi users are installed under the /usr/lpp/Wabi file tree. Files that are unique to each user such as, configuration files and installed applications are stored in $HOME/wabi or in a location pointed to by the users WABIDIR environment variable.

Listed below are some of the more significant files or directories that exist in both the Wabi and the users local file trees.

A.1 Files and Directories Installed under /usr/lpp/Wabi

bin	A directory containing Wabi executables
bin/dos2unix	Converts DOS format text files to UNIX format
bin/unix2dos	Converts UNIX format text files to DOS format
bin/wabi	Shell script that sets variables and paths, checks that the users $HOME/wabi directory exists and links the font cache, then calls wabiprog
bin/wabifs	Wabi font server - interacts with the X11R5 font server
bin/wabiprog	The Wabi program
wbin/integrate	For use in future desktop integration work
icons	A directory containing icons used for Wabi and the application items in the Wabi Tools panel
lib	A directory containing Wabi data files and default font caches for the various platforms supported by Wabi.
lib/ibm.dflt.fc	Default font cache for IBM systems
lib/locale	Keymaps and language specific DLLs for different locales
lib/wabifs.displays	Configuration file for the font server
man/man1/wabi.1	Wabi manual page
printers	A directory containing printer drivers
wbin	A directory containing the natively implemented versions of Windows programs, Wabi configuration files, and other files required to allow the user to emulate the Windows environment. These files will be copied or linked under the user's $HOME/wabi directory by the Wabi shell script the first time the user runs Wabi.

A.2 Files and Directories Installed under $HOME/wabi

autoexec.bat	The default autoexec.bat file is only used to set the PATH variable
fc	A directory where the user's font cache files are stored
tmp	A directory used for storing temporary files
wabihome	A symbolic link to directory /usr/lpp/Wabi
windows	The files that make up the user's Windows environment. The first time the user runs Wabi, the Wabi shell script creates this directory and copies or links some of the files from the /usr/lpp/Wabi/wbin tree to this directory. If the user installs Microsoft Windows 3.1 using the Windows Install icon, files are carefully extracted from the Windows installation media and interspersed with the files already here. This procedure is necessary to ensure that the natively implemented DLLs and files are not overwritten by the original Windows files of the same names.
windows/system	Windows system files. This corresponds to the system directory on Microsoft Windows 3.1.

Appendix B. Wabi 1.1 Manual Page

This is the manual page that is supplied with Wabi. It can be accessed by typing the command:

```
$ man wabi
```

at an AIX command prompt.

```
Wabi(1)                 (2 November 1993)                 Wabi(1)

NAME
     Wabi - Application to run Microsoft Windows-based
     applications

SYNOPSIS
     wabi [ program ] [ -s program ] [ -display display_name ]

DESCRIPTION
     Wabi allows you to run Microsoft Windows-based applications
     in a window on your UNIX-based workstation.

OPTIONS
     -s application_information
          Use "-s" to run the specified program without starting
          the shell specified in system.ini

     -display display_name
          Use "-display" to display Wabi on a remote display
          device.

EXAMPLES
     To start Wabi with the shell specified in system.ini:
               wabi

     To start Wabi with the shell and run an application:
               wabi C:/pathname/application_name.exe

     To start Wabi without the shell, and run an application:
               wabi -s C:/pathname/application_name.exe

     To start Wabi and display it to another display device:
               wabi -display hostname:0

ENVIRONMENT
     WABIHOME
          Location of Wabi executable files.  Set automatically
          when Wabi is started.

     WABIDIR
          Location of user's personal Wabi directory, $HOME/wabi
          by default.  Can be overridden by setting the WABIDIR
          environment variable.  When Wabi is invoked for the
          first time WABIDIR is created and default setup files
          are copied into this directory.
```

```
     DISPLAY
          Has same effect as "-display" option.  Sends Wabi
          output to another display device.

     WABI_KEYB
          Sets the keyboard language to affect the keystrokes
          Windows applications receive.  The default setting is
          "us".

     WABI_CODEPAGE
          Controls code page selection used by Windows for
          character translation.  Possible values are: 437
          (default), 850, 860, 861, 863, and 865.
     WABI_NOLOCK
          Set to a value of 1 to disable Wabi file and record
          locking.

SEE ALSO
     Wabi User's Guide
```

Appendix C. Redbook Sample Tools and Utility Programs

The following sample shell scripts, programs and data files may be used to assist you with setting up the Wabi environment. They are located in the RedbookTools directory. See C.1, “Installing the Redbook Sample Tools and Utilities” on page 112 for more information.

misc/fd2hd	This shell script can be used to copy install images from Windows and or Windows applications diskettes into an AIX directory that can be used to install the product.
misc/go_wabi	This shell script will start up a Wabi environment based on the LANG environment variable.
pcsim/simprof	This file is a sample AIX Personal Computer Simulator/6000 profile.
pcsim/dosexe	This shell script can be used to modify the AUTOEXEC.BAT file used by the AIX Personal Computer Simulator/6000 at startup to invoke your desired DOS application.
pcsim/translation.c	This is the C source code for a program used by the dosexe shell script mentioned above.
shared/give_attr	This shell script can be used to set up a Windows application to be used in a shared environment by setting read only permissions for the files and assigning them to a given AIX group.
shared/add_to_grp	This shell script can be used in the shared application environment to add a user to a given AIX group.
XDesktop/Wabi.obj	This is a sample XDesktop object that can be used to launch Wabi from the XDesktop.
XDesktop/xdtuserinfo_wabi	This file is a sample XDesktop rule file that can be used to set up a Windows application to be launched from the desktop.
XDesktop/wordicons	This directory contains sample icons that can be used with the above mentioned desktop rule file for launching Microsoft Word from the desktop.

Note: These items are samples and may or may not be completely functional in your particular environment. It is your responsibility to check over each item before using it to make sure that you fully understand what it is, what it is going to do and how it works. This is necessary, so that you can determine if it is going to do what you are expecting, and if any modifications will need to be made to make it work in your particular environment. The IBM corporation and the ITSO assume no responsibility for these items. They are provided only as samples on an AS IS basis. Use these samples at your own risk.

C.1 Installing the Redbook Sample Tools and Utilities

The following procedure can be used to install the Redbook Sample Tools and Utilities onto your AIX system:

1. Select the desired location to install the Redbook Sample Tools and Utilities. This procedure will create a directory called RedbookTools that contains the sample tools and utilities. You will need approximately 100KB of free disk space in the target installation filesystem.
2. Change directories using the `cd` AIX command to your target installation directory.
3. Insert the Redbook Sample Tools and Utilities diskette into the diskette drive on your system.
4. Enter the following AIX command to create a directory called RedbookTools and copy the sample tools and utilities into this directory:

   ```
   tar -xvf /dev/rfd0
   ```

5. Remove the diskette from the diskette drive.
6. Examine the dosexe utility and look for the lines identified by the comment "# **CHANGE THIS**". This comment marks places where you must modify the script to suit your specific environment.

C.2 Sample Tool/Utility: fd2hd

fd2hd is located in the /misc subdirectory of the RedbookTools directory.

The script copies the installation images for Microsoft Windows 3.1 and or Windows applications from diskette to a directory on your hard disk. This directory can then be NFS mounted to any workstations that need this application, and used instead of the installation diskettes to install the product.

The script first requests the number of diskettes to copy, then asks for the name of the directory to create. If it can successfully create the directory, it will then prompt the user for each diskette in turn and copy all files to the disk directory.

The syntax of this script is:

```
fd2hd
```

Script contents:

```
#! /usr/bin/ksh
###############################################################################
###                                                                         ###
###  Project:                   Wabi Redbook Tools                         ###
###  Module:                         fd2hd                                 ###
###  Authors :          Cameron Ferstat and Catherine Chevallier           ###
###  Date:                           17 Jan 94                             ###
###-------------------------------------------------------------------------###
###  Description: Utility shell script to grab all of the files from      ###
###               a series of diskettes and drop them all into a          ###
###               directory that will be created. Particularly useful     ###
###               for creating an install directory for Windows software  ###
###               from install diskettes.                                 ###
###############################################################################
```

```
typeset -i COUNT=1 NUMDISKS=0
typeset -l FILE
SRCDRIVE="/dev/fd0"
DESTDIR=""

MKDIR="/bin/mkdir"

# Get the number of diskettes to load.
print -n "Enter number of diskettes: "
read NUMDISKS

# Number of diskettes must be greater than zero.
if [[ $NUMDISKS -le 0 ]]
then
   print -u2 "Error: Number of diskettes must be greater than zero."
   exit 2
fi

# Get the destination directory.
print -n "Enter name of destination directory (default ./instdir): "
read DESTDIR ; DESTDIR=${DESTDIR:-"./instdir"}

# See if the destination directory exists. If not create it.
if [[ ! -d "$DESTDIR" ]]
then
   $MKDIR $DESTDIR
   if [[ $? -ne 0 ]]
   then
      # Error messages already given by mkdir
      exit 3
   fi
fi

# Go to the destination directory.
cd $DESTDIR

# For each one of the entered number of disketts, do a directory list and
# then do a dosread for each of the files on the diskette.
while [[ $COUNT -le $NUMDISKS ]]
do
   print -n "Insert diskette $COUNT in drive $SRCDRIVE and press <ENTER> "
   read

   for FILE in $(dosdir -a | grep -vi "free space")
   do
      case $FILE in
      ".")      ;;
      "..")     ;;
      *)        dosread $FILE $FILE ;;
      esac
   done

   COUNT=COUNT+1
done
```

C.3 Sample Tool/Utility: go_wabi

go_wabi is located in the /misc subdirectory of the RedbookTools directory.

The script will start up a Wabi environment based on the LANG environment variable.

The syntax of this script is:

```
go_wabi
```

Script contents:

```
#!/bin/ksh
#############################################################################
###                                                                       ###
###  Project:                   Wabi Redbook Tools                        ###
###  Module:                       go_wabi                                ###
###  Authors :          Cameron Ferstat and Catherine Chevallier          ###
###  Date:                           17 Jan 94                            ###
###-----------------------------------------------------------------------###
###  Description: Wabi utility shell script                               ###
###                                                                       ###
###               The shell script switches the WABI directory according ###
###               to the LANG variable environment and then runs Wabi.    ###
###                                                                       ###
#############################################################################

if [ $LANG = En_US ] ; then WABIDIR=$HOME/wabi_us ;export WABIDIR; fi
if [ $LANG = Fr_FR ] ; then WABIDIR=$HOME/wabi_fr ;export WABIDIR; fi

/usr/lpp/Wabi/bin/wabi
```

C.4 Sample Tool/Utility: simprof

simprof is located in the pcsim subdirectory of the RedbookTools directory.

This file is a sample AIX Personal Computer Simulator/6000 profile that will set up the PC Simulator with the following configuration.

- The 3.5 inch diskette drive is setup as the A: drive.
- The C: drive will be mapped to the AIX directory /u/cathy/dosdiskc. Remember to change this to be the directory or file for your C: drive.
- The D: drive will be mapped to the AIX directory /u/cathy/dosdiskd. Remember to change this to be the directory or file for your D: drive.
- The E: drive will be mapped to the AIX directory /u/cathy/dosdiske. Remember to change this to be the directory or file for your E: drive.
- DOS files will be stored in lowercase.
- The DOS printer will be mapped to the AIX printer queue `asc`. Remember to change this to be the name of your ASCII printer queue.
- The display will be refreshed every 50 milliseconds.
- 8MB of extended memory will be provided.
- The mouse data will be presented on the com1 serial port.

- VGA mode will be used for the display.

For more information see 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36.

File contents:

```
#Profile  simprof
#
#
Adiskette           : 3
Cdrive              : /u/cathy/dosdiskc
Ddrive              : /u/cathy/dosdiskd
Edrive              : /u/cathy/dosdiske
case                : L
lpt1                : asc
refresh             : 50
xmemory             : 8192
mouse               : com1
dmode               : V
```

C.5 Sample Tool/Utility: dosexe

dosexe is located in the pcsim subdirectory of the RedbookTools directory.

This is a sample shell script that can be used to modify the AUTOEXEC.BAT file that the AIX Personal Computer Simulator/6000 uses at startup, to invoke your desired DOS application. This script will call the translation.c program also included in this directory to assist with this task.

This is an example of an AUTOEXEC.BAT file that was updated with the shell script by issuing the command:

```
dosexe "D:\troispil\troispil.exe"

@echo off
path=c:\;c:\dos;
prompt $p$g
D:                                  # Added by dosexe
cd \troispil                        # Added by dosexe
troispil.exe                        # Added by dosexe
```

The syntax for this script is:

```
 dosexe doscommand [parameters]
```

Where:

doscommand Is the DOS command to execute after start up. This command must be a fully qualified path name and enclosed in quotes.

parameters If you have any parameters that need to be included on the above DOS command, include them as this parameter by enclosing all of them in a single set of quotation marks. This parameter is optional.

For more information see 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36.

Script contents:

```
#!/bin/ksh
###############################################################################
###                                                                         ###
###  Project:                  Wabi Redbook Tools                           ###
###  Module:                      dosexe                                    ###
###  Authors :          Cameron Ferstat and Catherine Chevallier            ###
###  Date:                          17 Jan 94                              ###
###-------------------------------------------------------------------------###
###  Description: Utility shell script that will update the                 ###
###               the AUTOEXEC.BAT file used by the PC Simulator            ###
###               at start up. The AUTOEXEC.BAT file is updated             ###
###               to execute the DOS command passed as a parameter          ###
###               to this script after the simulator starts up.             ###
###                                                                         ###
###               This script works by appending to the AUTOEXEC.BAT        ###
###               file the DOS commands necessary to switch DOS to the      ###
###               location of the desired command and then execute it.      ###
###                                                                         ###
###               When this script is run for the first time, a             ###
###               copy of the AUTOEXEC.BAT file is saved in a file          ###
###               named AUTOEXEC.SAV. And then the commands are             ###
###               appended to the contents of the AUTOEXEC.BAT file.        ###
###                                                                         ###
###               Subsequent executions of this script will append          ###
###               the commands to the contents of the saved version         ###
###               of the AUTOEXEC.BAT file located in the                   ###
###               AUTOEXEC.SAV file. This is done so that the               ###
###               script can be run multiple times without appending        ###
###               multiple groups of commands at the end of the             ###
###               AUTOEXEC.BAT file.                                        ###
###                                                                         ###
###               Be sure to put quotes around the paramters to this        ###
###               script. For example:                                      ###
###                  dosexe "D:\myappdir\myapp"  "parm1 parm2"              ###
###                                                                         ###
###               NOTE: You will have to modify this script to              ###
###               work in your environment. See the **CHANGE THIS**         ###
###               comments below.                                           ###
###############################################################################
# **CHANGE THIS**
# Change the following line to point to the location of the RedbookTools
# directory. This variable is used to locate the give_attr shell script.
WABITOOLS=$HOME/RedbookTools/pcsim

# **CHANGE THIS**
# Change the following lines to point to the location of the autoexec.bat and
# autoexec.sav files on the AIX directory that is your C: drive under the
# PC Simulator.
AUTOEXEC_BAT_SAV=$HOME/Cdrive/autoexec.sav
AUTOEXEC_BAT=$HOME/Cdrive/autoexec.bat

WABIDIR=$HOME/wabi
AIXBIN=/usr/bin
WABIBIN=$WABIDIR/wabihome/bin
REMAINDER_COMMAND=" "
```

```
typeset -l REMAINDER_COMMAND
typeset -l DOS_EXE
typeset -l DOS_PATH

# Usage
#
get_use()
{
    echo "\n Usage : dosexe <dos_command>  [  <parameters> ]\n"
    exit 1
}
#
#-------------------------------------------------------------------------
# Usage Test
# If only 1 parameter is passed, translate the path of the command
# If 2 parameters are passed, translate the path of the command & add parms
# Otherwise print the command usage
#
case $# in
        1) TR_PATH= $WABITOOLS/translation -d $1 | tr -d "'"   ;;
        2) TR_PATH= $WABITOOLS/translation -d $1 | tr -d "'"
           REMAINDER_COMMAND=$2;;
        *) get_use;;
esac
#-------------------------------------------------------------------------
# Extract the DOS diskdrive, command path and command from the passed DOS
# command name.
#
UNIX_PATH= dirname "${TR_PATH}"
DOS_PATH= $WABITOOLS/translation -u $UNIX_PATH
DOS_EXE= basename ${TR_PATH}
DISK_DRIVE= echo "${TR_PATH%:*}:"


# Test to see if there is already a dos program being run using this method.
if [ -f $AUTOEXEC_BAT_SAV ]
then
#  Yes: This is not the first run. Restore the original copy of AUTOEXEC.BAT
   $AIXBIN/cp $AUTOEXEC_BAT_SAV $AUTOEXEC_BAT
else
#  No: This is the first run. Save the original copy of AUTOEXEC.BAT
#  unless the AUTOEXEC.BAT file does not exist, then create an empty
#  one.
   if [ -f $AUTOEXEC_BAT ]
   then
      $AIXBIN/cp $AUTOEXEC_BAT $AUTOEXEC_BAT_SAV
   else
      $AIXBIN/touch $AUTOEXEC_BAT_SAV
   fi
fi
#  Append the DOS commands that will switch DOS to the location of the
#  desired command and then cause it to be executed.
   echo "$DISK_DRIVE\r" >> $AUTOEXEC_BAT
   echo "cd \c" >> $AUTOEXEC_BAT
   echo "${DOS_PATH#*:}\r" |tr -d "'"  >> $AUTOEXEC_BAT
   echo "$DOS_EXE \c" >> $AUTOEXEC_BAT
   echo "$REMAINDER_COMMAND\r" >> $AUTOEXEC_BAT
```

```
# Start up the PC Simulator
$AIXBIN/pcsim

# Copy back the old
$AIXBIN/cp $AUTOEXEC_BAT_SAV $AUTOEXEC_BAT
```

C.6 Sample Tool/Utility: translation.c

translation.c is located in the pcsim subdirectory of the RedbookTools directory.

This C source program is called by the above mentioned dosexe shell script to translate between DOS and AIX pathnames.

This program can be compiled by executing the following AIX command:

```
cc translation.c -o translation
```

For more information see 5.3.4, “Running DOS Applications from a Wabi Icon” on page 36.

Source program contents:

```
/*****************************************************************************
***                                                                        ***
***  Project:                  Wabi Redbook Tools                        ***
***  Module:                      translation.c                          ***
***  Authors :          Cameron Ferstat and Catherine Chevallier         ***
***  Date:                           24 Jan 94                           ***
***------------------------------------------------------------------------***
***  Description: Utility program that is called by dosexe to            ***
***               translate Dos Path to Unix Path (d option)             ***
***               or Unix Path to Dos Path (u option).                   ***
*****************************************************************************/

#include <string.h>
#include <stdio.h>
#include <stdarg.h>

#define MAX_PATH_LENGTH 1024
#define DOS_DELIMETER '\\'
#define UNIX_DELIMETER '/'

extern char *optarg;
extern int optind;
extern int opterr;
extern int optopt;


void fatal_error(char *format,...)
{
  va_list values;
  va_start(values,format);
  vfprintf(stderr,format,values);
  putchar('\n');
  exit(1);
}
```

```
/*
 * TranslateDosPath takes in a dos pathname specification
 * and converts it to a unix pathname
 * It converts the
 * backslashes (PATH_DELIMETER) to unix path delimeters ('/')
 */
int TranslateDosPath (char *dospath) {
        char    unixpath[MAX_PATH_LENGTH];
        int             i, l, lu;

        l = strlen(dospath);
        lu = 0;
        for (i=0; i<=l; i++) {
                if (dospath[i] != DOS_DELIMETER)
                        {
                         unixpath[lu++] = dospath[i];
                        }
                else
                        {
                         unixpath[lu++] = '/';
                        }
        }
        unixpath[lu++] = '\0';
printf("'%s'\n",unixpath);
}

/*
 * TranslateUnixPath takes in a Unix pathname specification
 * and converts it to a dos pathname
 * It converts the
 * unix path delimiters ('/') to dos delimiters ("\")
 */
int TranslateUnixPath (char *unixpath) {
        char    dospath[MAX_PATH_LENGTH];
        int             i, l, lu;

        l = strlen(unixpath);
        lu = 0;
        for (i=0; i<=l; i++) {
                if (unixpath[i] != UNIX_DELIMETER)
                        {
                         dospath[lu++] = unixpath[i];
                        }
                else
                        {
                         dospath[lu++] = '\\';
                        }
        }
        dospath[lu++] = '\0';
printf("'%s'\n",dospath);

}

void usage(char *progname)
{
  printf("Usage : %s -d dos-path"
             "   -u unix-path"
```

```
         "\n",progname);
}

/* main */
int main (int argc, char **argv) {
        int c;
        char    *dos_path;
        char    *unix_path;


  /* Line option decoding */
  while((c=getopt(argc,argv,":hd:u:"))!=EOF)
  {
    switch(c)
    {
    case 'h' : usage(argv[0]); exit(0); break;
    case 'd' : dos_path=optarg;
               if (dos_path==NULL) fatal_error("No path name");
               else TranslateDosPath(dos_path); break;
    case 'u' : unix_path=optarg;
               if (unix_path==NULL) fatal_error("No path name");
               else TranslateUnixPath(unix_path); break;
    case '?' : fatal_error("Unknow option -%c\n",optopt); break;
    case ':' : fatal_error("Missing argument for option -%c",optopt); break;
    }
  }
  /* We check that all needed options are here */
  if(optind<argc) fatal_error("Error : remaining arguments");
}
```

while((c=getopt(argc,argv,":hd:u:"))!=EOF)

C.7 Sample Tool/Utility: give_attr

give_attr is located in the shared subdirectory of the RedbookTools directory.

This shell script can be used to set up a Windows application to be used in a shared environment by setting read only permissions for the files and assigning them to a given AIX group.

Note: You must be the root user to run this script.

The syntax for this script is:

```
give_attr appdir group
```

Where:

appdir Is the path name of the AIX directory that contains the application that is to be set up in a shared environment. This directory, it's sub-directories and all of the files in them will be set to read only permissions

group This is the name of an AIX group that you want the application in the above directory to be owned by

For more information see 8.2.4.2, “File Permissions” on page 76.

Script contents:

```
#!/bin/ksh
###############################################################################
###                                                                         ###
###  Project:                   Wabi Redbook Tools                          ###
###  Module:                       give_attr                                ###
###  Authors :          Cameron Ferstat and Catherine Chevallier            ###
###  Date:                           17 Jan 94                              ###
###-------------------------------------------------------------------------###
###  Description: Utility shell script that can be used to setup            ###
###               a Windows application to be used in a shared              ###
###               environment. This script will set read only               ###
###               permissions to all of the directories and                 ###
###               files that are part of the application so that            ###
###               they can not be modified. Also, the directories           ###
###               and files will be set to be owned by the group            ###
###               ID that is passed as the second parameter.                ###
###                                                                         ###
###               You must be the root user to run this script              ###
###                                                                         ###
###############################################################################
#------------------------------------------------------------------------------
# Test for the correct number of parameters. If incorrect, print usage.
#
  if [ $# -ne 2 ]
   then
    echo "\nUsage : give_attr directory group_name\n"
    exit 1
  fi

#------------------------------------------------------------------------------
# Test to make sure that this script was run as root. If not error and exit
#
if [  whoami  != "root" ];
then
 echo "\nYou must be root to run this script!\n"
 exit 1
fi

#------------------------------------------------------------------------------
AIXBIN=/usr/bin

#This first section of the script changes the application to be
#owned by the passed group name.
#First check to see if the directory exists.
if [ -d $1 ]
then
# The directory exists.
# Check to see if the group exists.
  $AIXBIN/lsgroup $2 >/dev/null 2>&1
  if [ $? -ne 0 ]
  then
#    The group did not exist. Add it.
     $AIXBIN/mkgroup $2
  fi
# The directory and the group exist; change the files to be owned by
# the group.
```

```
  $AIXBIN/chgrp -R $2 $1
else
# The directory does not exist. Print error message and exit.
  echo "$1: application doesn't exist"
  exit 1
fi

#This next section sets the file/directory permissions to be read only.
find $1 -print | {
   while read PATHNAME
   do
      if [[ -d $PATHNAME ]]
      then
         $AIXBIN/chmod ug+rx,g-w,o-rwx $PATHNAME
      else
         $AIXBIN/chmod g-w,o-rwx $PATHNAME
      fi
   done
}
```

C.8 Sample Tool/Utility: add_to_grp

add_to_grp is located in the shared subdirectory of the RedbookTools directory.

This shell script can be used to add a user's id to the specified group

Note: You must be the root user to run this script.

The syntax for this script is:

```
add_to_grp userid group
```

Where:

userid Is the AIX userid that is to be added to the group.

group This is the name of an AIX group that you want to add the userid in.

For more information see 8.2.4.2, “File Permissions” on page 76.

Script contents:

```
#!/bin/ksh
#############################################################################
###                                                                       ###
###  Project:                  Wabi Redbook Tools                         ###
###  Module:                      add_to_grp                              ###
###  Authors :          Cameron Ferstat and Catherine Chevallier          ###
###  Date:                           17 Jan 94                           ###
###-----------------------------------------------------------------------###
###  Description: Utility shell script that adds the passed userid        ###
###               to the passed group.                                    ###
###                                                                       ###
###               You must be the root user to run this script            ###
###               or an administrator of the group                        ###
#############################################################################
#----------------------------------------------------------------------------
```

```
# Test for the correct number of parameters. If incorrect, print usage.
#
  if [ $# -ne 2 ]
   then
    echo "\n Usage : add_to_grp  user_name group_name\n"
    exit 1
  fi

#--------------------------------------------------------------------------

USER=$1
WABI_GRP=$2
AIXBIN=/usr/bin

#--------------------------------------------------------------------------
# Add a group in the user's group list and capture the results

RETURN=$(chgrpmem -m + $USER $WABI_GRP 2>&1)
RETCODE=$?

# Display results a bit more helpfully than chgrpmem
case $RETCODE in
0)      echo "User \"$USER\" has been added to group \"$WABI_GRP\"" ;;
1)      echo "You do not have permission to run this command."
        echo "You must be root, or an administrator of group \"$WABI_GRP\"" ;;
2)      echo $RETURN ;;
esac

exit $RETCODE
```

C.9 Sample Tool/Utility: Wabi.obj

Wabi.obj is located in the XDesktop subdirectory of the RedbookTools directory.

Wabi.obj is an XDesktop object that will launch Wabi from the desktop.

For more information see 7.4.1, “AIX Desktop” on page 60.

Included in this object is several icons that are used by this object and the following two trigger action scripts:

Script contents:

```
#########################################################################
###                                                                   ###
###  Project:           Wabi Redbook Examples                         ###
###  Module:                 d1.objscr                                ###
###  Author:                Mark Kressin                              ###
###  Organization:     ITSO - Austin: IBM Corp.                       ###
###  Date:                  16 March 94                               ###
###-------------------------------------------------------------------###
### Description: This is a sample AIX Desktop* script that is invoked  ###
###              when a user drops a file onto the Wabi** object.      ###
###              This script will verify that the dropped file is a    ###
###              .exe application file and then use Wabi to start      ###
###              the application.                                      ###
### ------------------------------------------------------------------###
```

```
###  *  Copyright IBM Corporation                                        ###
###  ** Copyright Sun Microsystems Incorporated                         ###
###                                                                     ###
#########################################################################
for idx in $dynamic_args
  do
    if [ =  ( extension $idx ) .exe ]
    then
      /usr/lpp/Wabi/bin/wabi -s  $idx
    else
      fyi $idx is not a Windows or DOS executable
    fi
  done




#########################################################################
###                                                                     ###
###  Project:           Wabi Redbook Examples                           ###
###  Module:                 s1.objscr                                  ###
###  Author:                Mark Kressin                                ###
###  Organization:     ITSO - Austin: IBM Corp.                         ###
###  Date:                  16 March 94                                 ###
###---------------------------------------------------------------------###
### Description: This is a sample AIX Desktop* script that is invoked  ###
###              when a user double clicks on the Wabi** object icon.  ###
###              This script will start Wabi.                           ###
### --------------------------------------------------------------------###
###  *  Copyright IBM Corporation                                        ###
###  ** Copyright Sun Microsystems Incorporated                         ###
###                                                                     ###
#########################################################################
/usr/lpp/Wabi/bin/wabi
```

C.10 Sample Tool/Utility: xdtuserinfo_wabi

xdtuserinfo_wabi is located in the XDesktop subdirectory of the RedbookTools directory.

xdtuserinfo_wabi is an example XDesktop rule file that integrates the Microsoft Word application and Wabi into the desktop. This rule file will tell the desktop to do the following:

- Display a Word icon for the Word executable
- Allow the user to double click on the Word icon to bring up the Word application
- Allow the user to drop a Word document file on the Word executable to have Word started with the dropped document loaded
- Display a Word document icon for each .doc file
- Allow the user to double click on a Word document to have Word started and the selected document loaded

For more information see 7.4.1, “AIX Desktop” on page 60.

Rule file contents:

```
%//##########################################################################
%//###                                                                    ###
%//###  Project:           Wabi Redbook Examples                          ###
%//###  Module:              xdtuserinfo_wabi                             ###
%//###  Author:                Mark Kressin                               ###
%//###  Organization:     ITSO - Austin: IBM Corp.                        ###
%//###  Date:                  16 March 94                                ###
%//###------------------------------------------------------------------###
%//### Description: This is a sample AIX Desktop* user rule file that     ###
%//###              integrates the Microsoft Word** application and       ###
%//###              Wabi*** into the desktop. This rule file will tell    ###
%//###              the desktop to do the following:                      ###
%//###                o Display a Word icon for the Word executable       ###
%//###                o Allow the user to double click on the Word        ###
%//###                  icon to bring up the Word application             ###
%//###                o Allow the user to drop a Word document file       ###
%//###                  on the Word executable to have Word started with  ###
%//###                  the dropped document loaded                       ###
%//###                o Display a Word document icon for each .doc file   ###
%//###                o Allow the user to double click on a Word          ###
%//###                  document to have Word started and the selected    ###
%//###                  document loaded.                                  ###
%//### ----------------------------------------------------------------###
%//###  *   Copyright IBM Corporation                                     ###
%//###  **  Copyright Microsoft Corporation                               ###
%//###  *** Copyright Sun Microsystems Incorporated                       ###
%//###                                                                    ###
%//##########################################################################
icon_rules
{
  %/* Rule for Word executable file %*/
  winword.exe /F
  {
   title=Microsoft Word;
   picture=winword.px;
   %/* Double click action: Start Wabi and then Word %*/
   trigger_action : s1
    {
     /usr/lpp/Wabi/bin/wabi -s $static_arg
    }
   %/* File drop action: If file is .doc. Start Wabi and Word then load .doc %*/
   trigger_action : d1
    { for i in $*
    do
      if [ =  ( extension $i ) .doc ]
      then
         /usr/lpp/Wabi/bin/wabi -s $static_arg 'r:'$i
      else
        fyi $i is not a Word document file
      fi
    done
    }
  }
  %/* Rule for Word document file %*/
  *.doc /F
  {
```

```
   picture=windoc.px;
   %/* Double click action: Start Wabi and Word then load .doc %*/
   trigger_action : s1
    {
     /usr/lpp/Wabi/bin/wabi -s $HOME/wabi/winword/winword.exe 'r:'$static_arg
    }
  }
}
```

C.11 Sample Tool/Utility: wordicons

wordicons is located in the XDesktop subdirectory of the RedbookTools directory.

wordicons is a directory that contains the following example icons for the Microsoft Word application. These icons are used by the xdtuserinfo_wabi rule file listed above.

- windoc.px - Microsoft Word document icon.
- winword.px - Microsoft Word application icon.

For more information see 7.4.1, “AIX Desktop” on page 60.

List of Abbreviations

386	80386 Microprocessor
AFS	Andrew File System
AIX	AIX/6000
API	Application Programming Interface
CAD	Computer Aided Design
CD_ROM	Compact Disk - Read Only Memory
DDE	Dynamic Data Exchange
DLL	Dynamically Linked Library
DOS	Disk Operating System
IBM	International Business Machines Corporation
I/O	Input/Output
ITSO	International Technical Support Organization
LPP	Licensed Program Product
NLS	Native Language Support
NFS	Network File System
OLE	Object Linking and Embedding
OSF	Open Software Foundation
PC	Personal Computer
PCSIM	AIX Personal Computer Simulator/6000
PID	Process Identifier
PTF	Program Temporary Fix
RISC	Reduced Instruction Set Computer
SMIT	System Management Interface Tool
Windows	Microsoft Windows 3.1
x86	The family of Intel microprocessors, including 8086, 80286, 80386, 80486.

Index

R

S

T

U

V

W

X

ITSO Technical Bulletin Evaluation
RED000

IBM User's Guide to Wabi

Publication No. GG24-4304-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction	____		
Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name	Address
Company or Organization	
Phone No.	

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 948, Building 821
Internal Zip 2834
11400 BURNET ROAD
AUSTIN TX
USA 78758-3493

Fold and Tape **Please do not staple** Fold and Tape

Cut or Fold
Along Line

Printed in U.S.A.